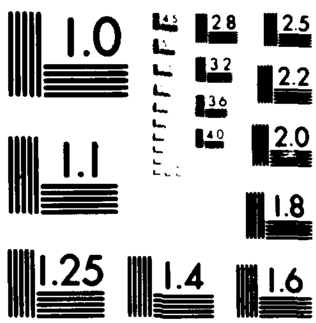
AD-A090 965 TRW DEFENSE AND SPACE SYSTEMS GROUP REDONDO BEACH CA F/6 9/2
AIRBORNE SYSTEMS SOFTWARE ACQUISITION ENGINEERING GUIDEBOOK FOR--ETC(U)
SEP 78 F33657-76-C-0677
UNCLASSIFIED TRW-30323-6013-TU-00 ASD-TR-79-5027 NL

1 of 1

END
DATE
FILMED
12-80
DTIC



MICROCOPY RESOLUTION TEST CHART

LEVEL II

ASD-TR-79-5027

1

Airborne Systems
Software Acquisition Engineering Guidebook
for
REQUIREMENTS ANALYSIS AND SPECIFICATION

AD A090965

SEPTEMBER 1978

PREPARED FOR
DEPUTY FOR ENGINEERING
AERONAUTICAL SYSTEMS DIVISION
WRIGHT-PATTERSON AFB, OH 45433

PREPARED BY
TRW DEFENSE AND SPACE SYSTEMS GROUP
ONE SPACE PARK
REDONDO BEACH,CA 90278

NOTICE

When Government drawings, specifications, or other data are used for any purpose other than in connection with a definitely related Government procurement operation, the United States Government thereby incurs no responsibility nor any obligation whatsoever; and the fact that the government may have formulated, furnished, or in any way supplied the said drawings, specifications, or other data, is not to be regarded by implication or otherwise as in any manner licensing the holder or any other person or corporation, or conveying any rights or permission to manufacture use, or sell any patented invention that may in any way be related thereto.

This report has been reviewed by the Office of Public Affairs (ASD/PA) and is releasable to the National Technical Information Service (NTIS). At NTIS, it will be available to the general public, including foreign nations.

This technical report has been reviewed and is approved for publication.

JOHN M. HOEFERLIN, Project Engineer
Information Engineering Division

RICHARD J. SYLVESTER
ASD Computer Resources Focal Point

FOR THE COMMANDER

ROBERT P. LAVOIE, Colonel, USAF
Director of Avionics Engineering
Deputy for Engineering

"If your address has changed, if you wish to be removed from our mailing list, or if the addressee is no longer employed by your organization please notify ASD/ENAIA, W-PAFB, OH 45433 to help us maintain a current mailing list".

Copies of this report should not be returned unless return is required by security considerations, contractual obligations, or notice on a specific document.

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

19 REPORT DOCUMENTATION PAGE

READ INSTRUCTIONS BEFORE COMPLETING FORM

1. REPORT NUMBER	18 ASD-TR-79-5027
2. GOVT ACCESSION NO.	AD-A090 965
3. RECIPIENT'S CATALOG NUMBER	
4. TITLE (and Subtitle)	6 Airborne Systems Software Acquisition Engineering Guidebook for Requirements Analysis and Specification.
5. TYPE OF REPORT & PERIOD COVERED	9 FINAL rept.
6. PERFORMING ORG. REPORT NUMBER	14 TRW-30323-6013-TU-00
7. AUTHOR(s)	
8. CONTRACT OR GRANT NUMBER(s)	15 F33657-76-C-0677
9. PERFORMING ORGANIZATION NAME AND ADDRESS	TRW Defense and Space Systems Group One Space Park Redondo Beach, CA 92078
	11 Sep 78
10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS	PE64740F Project 16 2238
11. CONTROLLING OFFICE NAME AND ADDRESS	Hq ASD/ENAIA Wright-Patterson AFB OH 45433
	12 77
12. REPORT DATE	September 1978
13. NUMBER OF PAGES	70
14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office)	
15. SECURITY CLASS. (of this report)	UNCLASSIFIED
15a. DECLASSIFICATION/DOWNGRADING SCHEDULE	

16. DISTRIBUTION STATEMENT (of this Report)

Approved for Public Release
Distribution Unlimited

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

Software Requirements Analysis, Software Requirements Specification, Software Development Specification

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

This report is one of a series of guidebooks which provide guidance to the acquisition management and engineering of Airborne Systems software procured under Air Force 800-series regulations. It describes the derivation, analysis, and documentation of airborne software requirements.

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

409637

PREFACE

This guidebook is one of a series of guidebooks intended to assist Air Force Program Office and Engineering personnel in software acquisition engineering for airborne systems. The contents of the guidebooks will be revised periodically to reflect changes in software acquisition policies and practices and feedback from users.

This guidebook was prepared under the direction of the Aeronautical Systems Division, Deputy for Engineering (ASD/EN) in coordination with the Space Division, Deputy for Acquisition Management (SD/AQM).

The entire series of Software Acquisition Engineering Guidebooks (Airborne Systems) is listed below along with ASD Technical Report numbers and NTIS accession numbers where available.

Regulations, Specifications and Standards	ASD-TR-78-6	ADA058428
Reviews and Audits	ASD-TR-78-7	ADA058429
Software Quality Assurance	ASD-TR-78-8	ADA059068
Configuration Management	ASD-TR-79-5024	ADA076542
Computer Program Documentation Requirements	ASD-TR-79-5025	ADA076543
Statements of Work and Requests for Proposal	ASD-TR-79-5026	ADA076544
Requirements Analysis and Specification	ASD-TR-79-5027	
Verification, Validation and Certification	ASD-TR-79-5028	
Microprocessors and Firmware	ASD-TR-80-5021	
Software Development Planning and Control	ASD-TR-80-5022	
Software Testing and Evaluation	ASD-TR-80-5023	
* Contracting for Software Acquisition	ASD-TR-80-5024	

* **Software Cost Analysis and Estimating** — **ASD-TR-80-5025**
* **Supportable Airborne Software** — **ASD-TR-80-5026**
* **Software Development and Support Facilities** — **ASD-TR-80-5027**
* **SAE Guidebooks - Application and Use** — **ASD-TR-80-5028**

*** These Guidebooks Available Fall 1980.**

CONTENTS

CONTENTS (Continued)

CONTENTS (Concluded)

ILLUSTRATIONS

TABLES

ABBREVIATIONS AND ACRONYMS

ACS	Armament Control Set
AFLC	Air Force Logistics Command
AFM	Air Force Manual
AFP	Air Force Pamphlet
AFR	Air Force Regulation
AFSC	Air Force System Command
AFSCP	Air Force System Command Pamphlet
AHRS	Attitude Heading Reference Set
ANSI	American National Standard Institute
ARSP	Analog Radar Signal Processor
AS	Airborne Software
ASP	Avionic Status Panel
BIT	Built-In Test
BITCP	BIT Control Panel
CC	Central Computer
CCA	Central Computer Assembler
CDR	Critical Design Review
CI	Configuration Item
CPCI	Computer Program Configuration Item
CPIN	Computer Program Identification Number
DH	Design Handbook
DODD	Department of Defense Directive
DODI	Department of Defense Instruction
DODM	Department of Defense Manual
FIPS PUB	Federal Information Processing Standards Publication
FSED	Full Scale Engineering Development
F-X	Fighter-X
F-X AS	F-X Airborne Software
HSI	Horizontal Situation Indicator
HUD	Heads-Up Display

ABBREVIATIONS AND ACRONYMS (Concluded)

INS	Inertial Navigation System
LCG	Lead Computing Gyro
LDR	Life Data Recorder
MENS	Mission Element Need Statement
MIL	Military
MIL-S	Military Specification
MPI	Multipurpose Indicator
MUX	Multiplex
NCI	Navigation Control Indicator
OFP	Operational Flight Program
RDPS	Radar Data Processing Software
SAE	Software Acquisition Engineering
SAMSO	Space and Missile Systems Organization
SON	Statement of Operational Need
STD	*Standard*

1. INTRODUCTION

1.1 PURPOSE AND SCOPE

The purpose of this guidebook is to assist Air Force program office engineering personnel in the derivation, analysis, and documentation of airborne software requirements for inclusion in a Request for Proposal (RFP) for Full Scale Engineering Development (FSED) of a weapon system or a portion of a weapon system. The airborne software may be for a totally new weapon system, or for a modification/retrofit of an existing weapon system. In the latter case, the task of deriving and specifying airborne software requirements may be simplified because the weapon system hardware characteristics may be better defined and because some requirements on the software may already exist. Hence, the task may reduce to modifying existing software requirements to reflect the objectives of the modified weapon system. In either case, the guidelines presented here should be helpful in systematically deriving, analyzing, and documenting a complete set of airborne software requirements for inclusion in the RFP. The concepts presented here could be applied to firmware for microprocessors as well as to conventional software. Generally, the software requirements are included in the Statement of Work (SOW) portion of the RFP, either directly or by reference to a separate document that contains the software requirements.

This guidebook addresses the engineering activity that occurs during the validation phase of the weapon system life cycle. It provides information concerning the allocation, generation and authentication of airborne software requirements based on the weapon system requirements available at the final System Requirements Review (SRR) and documented in the preliminary system specification. This guidebook does not, however, describe how to produce a complete Computer Program Part I Development Specification; it does not address the software testing requirements that go into the Quality Assurance section, Section 4, of a Computer Program Part I Development Specification. Software testing is discussed in the SAE Guidebook for Software Quality Assurance and in the SAE Guidebook for Verification, Validation, and Certification. This guidebook does address

all the requirements that would appear in Section 3 of a Part I specification and, hence, may be useful during preparation of a Part I specification. In some cases the airborne software requirements in the RFP are referred to as a Preliminary Computer Program Part I Development Specification, i.e., where the airborne software is a single Computer Program Configuration Item (CPCI). In general, however, the airborne software addressed in an RFP may consist of several CPCI's. For example, the weapon system may include several airborne digital processors, each utilizing a separate CPCI. In this case the RFP software requirements for all the airborne software may be documented jointly or they may be documented in separate preliminary Part I development specifications for the CPCI's that make up the airborne software. This guidebook is applicable in either case.

1.2 PURPOSE OF SOFTWARE REQUIREMENTS IN A REQUEST FOR PROPOSAL

The purpose of airborne software requirements in an RFP is to specify to the bidders all the known requirements of the software which executes in the airborne digital processors of the weapon system or portion of the weapon system addressed by the RFP. Definitive software requirements are essential to ensure airborne software which will accomplish the mission objectives of the weapon system and will also provide for expansion to accomplish future new objectives of the system while minimizing the life cycle cost of the weapon system. The software requirements derived from system specifications establish functional, performance, and interface requirements that the airborne software must be designed to meet, and establish development standards and constraints to which the airborne software and its documentation must comply. Good software requirements as part of the Request for Proposal convey to the potential contractor precisely what the Air Force requires. This enables the potential contractor (bidder) to define a viable technical approach and arrive at an accurate estimate of the software development cost. It also increases the probability that the end product will satisfy Air Force objectives. Properly defined software requirements serve as a basis for the progressive definition and generation of CPCI Part I development specifications and can also serve as a basis for airborne software design and development after the contract is awarded.

Experience shows that good requirements early during software development pays off in dollars and better software performance and also facilitates top-down design, meaningful testing, and management visibility and control.

Software requirements analysis is a prerequisite for specifying software requirements in the RFP. A thorough analysis of objectives and interfaces is necessary to derive and document the specific and quantitative airborne software requirements.

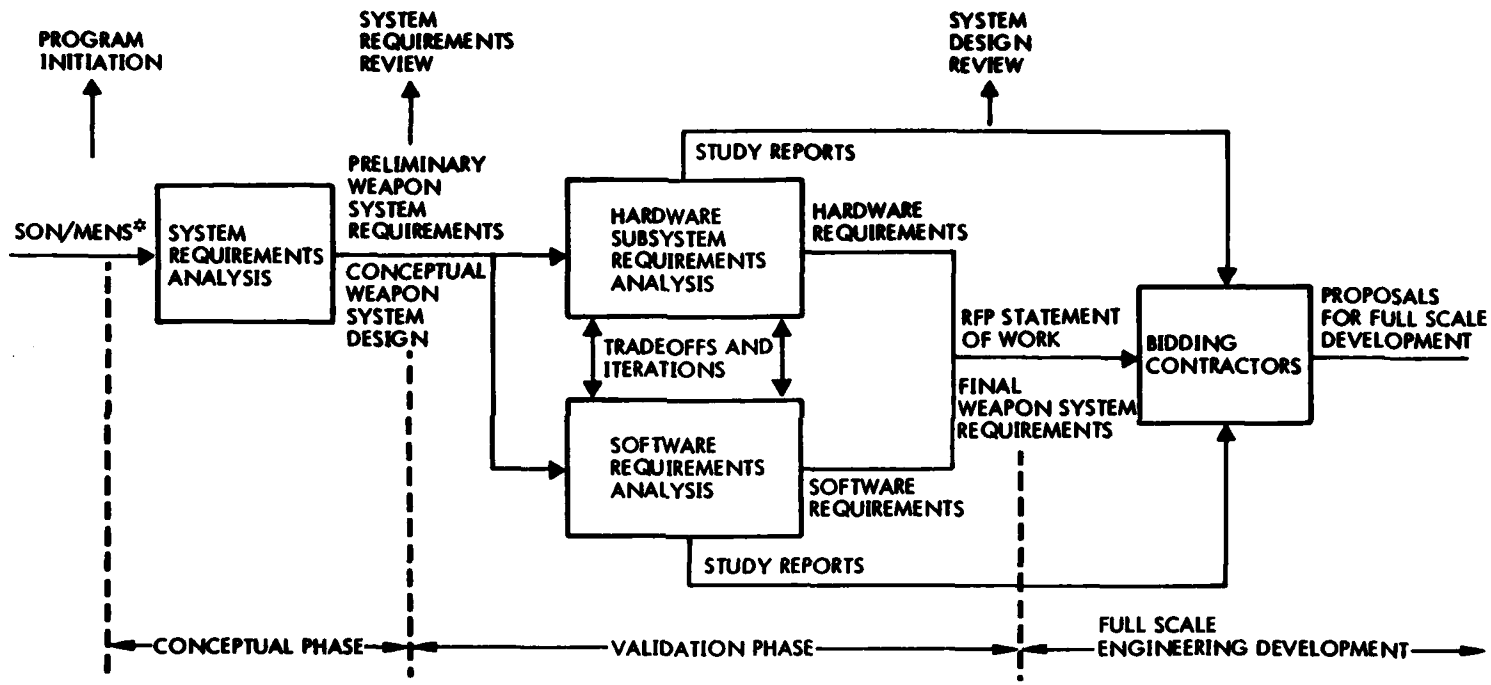
1.3 GENERAL DESCRIPTION OF SOFTWARE REQUIREMENTS

The airborne software requirements in the RFP should include all requirements that the airborne software must satisfy to perform its function adequately. The requirements are based on the weapon system level requirements and on requirements analysis. The airborne software requirements in the RFP describe the functions that the airborne software must perform, and they include quantitative performance requirements that specify how well it performs these functions. The requirements also define the environment in which the airborne software must operate by describing the interfaces between it and other elements[†] of the weapon system and by describing the performance characteristics of the airborne system to the extent that they can effect the airborne software performance. Finally, the requirements may include constraints and standards that must be followed during airborne software design and development.

1.4 GENERAL DESCRIPTION OF REQUIREMENTS ANALYSIS

Figure 1-1 depicts the system engineering process that results in software requirements that go into the FSED RFP. The system engineering process involves a hierarchy of requirements generation, beginning with an operational mission need and ending with detailed engineering specifications and data. Each level of requirements (beginning with functional requirements) leads to the next lower level of detail until the entire system is specified. The process continues to define and optimize the requirements on subsystems that make up the total weapon system.

[†] In this document, the crew on an airborne system is considered to be an element or a subsystem in that weapon system.



*SON - STATEMENT OF OPERATIONAL NEED
MENS - MISSION ELEMENT NEED STATEMENT, FORMERLY CALLED A REQUIRED OPERATIONAL CAPABILITY (ROC)

Figure 1-1. Software Requirements Analysis

This engineering data is then analyzed to determine design requirements, and engineering and tradeoff studies are performed to determine the best approach to a total weapon system configuration. This effort results in a system specification and the establishment of a functional baseline for the weapon system.

These system level requirements are allocated to hardware† and software based on system performance and life cycle cost considerations. The airborne software requirements derived from the system specification must then undergo similar engineering analysis and tradeoffs to determine functional, performance, interface and test requirements. Documentation of the requirements generated in this iterative process is then provided to potential system or subsystem developers as part of the Request for Proposal, most often in the form of the Part I development specification. This documentation can aid the developing contractor in preparing the final Part I specification which, when authenticated,‡ becomes the allocated baseline. This is the basis for full scale engineering development.

A major task for the program office software engineer is to direct, monitor and participate in this engineering effort that culminates in authentication of Part I specifications.

1.5 CONTEXT OF SOFTWARE REQUIREMENTS ANALYSIS

1.5.1 Within the System Life Cycle

Software requirements analysis is initiated when system requirements analysis on a weapon system determines that computer equipment and computer programs are appropriate solutions to design objectives. Since the software requirements are a part of a FSED Request for Proposal, the software requirements analysis and specification activities must occur before the FSED Request for Proposal is issued; i.e., software requirements analysis and specification must take place during the validation phase of system development, while the hardware requirements analysis is taking

†In this guidebook, the term "hardware" refers to digital processing equipment and any other non-software elements in the weapon system.

‡Paragraph 3.4.2 of MIL-STD 483 states that this authentication process must be accomplished within 90 days after contract award.

place. Figure 1-2 depicts how software requirements generation fits into the life cycle of the airborne system. The software requirements in the RFP are used by the bidders during proposal generation, and are used by the winning contractor in completing the CPCI Part I development specifications for the airborne software. The software requirements may also be used in preparing an airborne software integration test plan and in evaluating the airborne software test results. Information derived from the software requirements analysis activity is included in the Computer Program Development Plan and may also be useful in determining support requirements and responsibilities to assist in preparation of the Computer Resources Integrated Support Plan. The concepts set forth in this guidebook are consistent with the management principles specified in AFR 800-14 and MIL-STD-499A.

1.5.2 Within the Guidebook Series

While this guidebook addresses the derivation and documentation of airborne software requirements, the other Software Acquisition Engineering Guidebooks address how those requirements will be used and how the airborne software will be developed, reviewed, and tested to assure that the end product satisfies the requirements and has been demonstrated to do so. This guidebook contains references to other software acquisition engineering guidebooks for amplifying information and applications when appropriate.

1.6 GUIDEBOOK CONTENTS

This guidebook contains the following parts:

- Abbreviations and Acronyms. A list of abbreviations and acronyms used in the guidebook.

- Section 1, Introduction. Describes the purpose and scope of this guidebook, states the context for software requirements analysis and specification and outlines the contents of this guidebook.

- Section 2, Relevant Documents. Lists Department of Defense and Air Force documents that are particularly relevant to this guidebook.

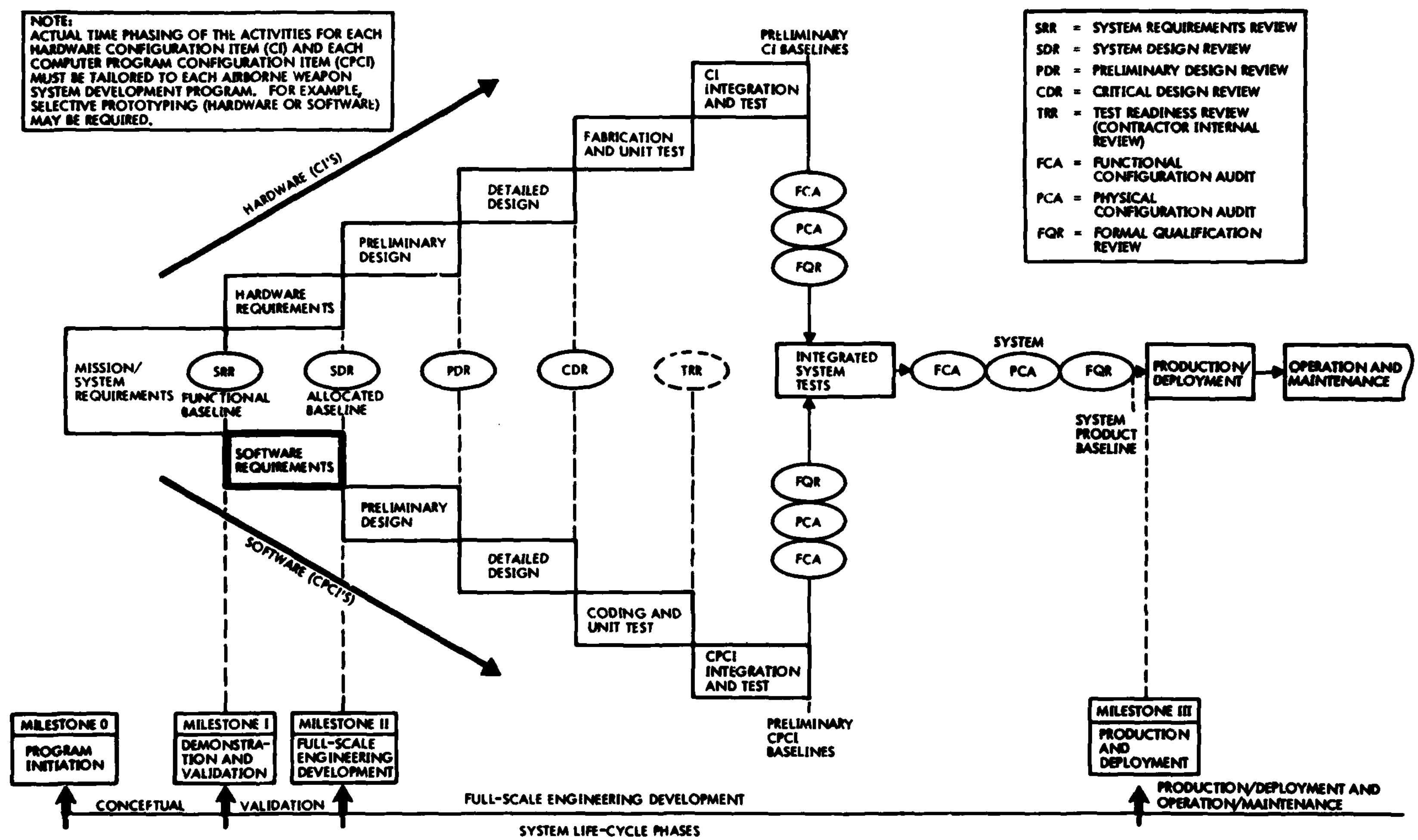


Figure 1-2. Idealized System Life Cycle

- <u>Section 3, General Guidelines</u>. Presents general guidelines and procedures for analyzing and specifying airborne software requirements. Identifies the set of requirements associated with software procurement and discusses assessment of those requirements relative to completeness, traceability, testability, consistency, and feasibility.

- <u>Section 4, Software Functional and Performance Requirements</u>. Discusses each type of airborne software functional and performance requirement, gives the purpose of the requirement, and describes how to derive and document the requirement.

- <u>Section 5, Software Development Standards and Constraints</u>. Discusses the various types of software development standards and constraints that may be appropriate to include in an RFP, and describes how to select and document those standards and constraints.

2. RELEVANT DOCUMENTS

The following list of documents are particularly relevant to this guidebook; they are referenced in this guidebook and, therefore, become an extension of this guidebook.

2.1 GOVERNMENT DOCUMENTS PERTAINING TO ACQUISITION MANAGEMENT PRACTICES

1.	AFR 57-1	Operational Requirements Statement of Operational Need (SON)
2.	AFR 800-14	Management of Computer Resources in Systems (Volumes I and II)

2.2 GOVERNMENT DOCUMENTS PERTAINING TO MANAGEMENT CONTROLS

1.	MIL-STD 480†	Configuration Control-Engineering Changes, Deviations, and Waivers
2.	MIL-STD 483	Configuration Management Practices for Systems, Equipment, Munitions, and Computer Programs
3.	MIL-STD 490	Specifications Practices
4.	MIL-STD 1521A	Technical Reviews and Audits for Systems, Equipment, and Computer Programs
5.	MIL-S 52779	Software Quality Assurance Program Requirements
6.	MIL-S 83490	Specifications, Types and Forms

2.3 GOVERNMENT DOCUMENTS PERTAINING TO HUMAN ENGINEERING

1.	AFR 80-46	Management of Personnel Subsystem/ Human Factors in System, Subsystem, Equipment, and Modification Development
2.	AFR 800-15	Human Factors Engineering Management
3.	AFSC DH 1-3	Personnel Subsystem Design Handbook

†MIL-STD-480 is expected to be replaced by DOD-STD 480A.

4.	MIL-STD 1472B	Human Engineering Design Criteria for Military Systems Equipment and Facilities
5.	MIL-H 46855A	Human Engineering Requirements for Military Systems, Equipment, and Facilities

2.4 GOVERNMENT DOCUMENTS PERTAINING TO SAFETY

1.	AFR 127-8	Responsibilities for USAF System Safety Engineering Programs (AFSC Supplement, 4/11/73)
2.	AFR 127-13	Responsibilities for the USAF Aerospace Safety Program (AFSC Supplement, 8/12/74)
3.	MIL-STD 454	Standard General Requirements for Electronic Equipment
4.	MIL-STD 882A	Requirement for System Safety Program for Systems and Associated Subsystems and Equipment

2.5 GOVERNMENT DOCUMENTS PERTAINING TO SECURITY

1.	AFM 207-1	Doctrine and Requirements for Security of Air Force Weapons Systems
2.	AFM 207-3	Aircraft Systems Security Standards
3.	AFM 207-21	System Security Standard-Command and Control and Communication System (Reprint, 7/24/74)
4.	AFP 205-2	Communications Security and Transmission Security
5.	AFR 8-9	USAF Communications Security and Emanations Security Publications
6.	AFR 80-7	Communications Security Research, Development, Test, and Evaluation Procedures (AFSC Supplement, 10/5/71)
7.	AFR 100-27	Release or Disclosure of Unclassified Messages
8.	AFR 100-51	Control of Compromising Emanations (TEMPEST) (ESD Supplement, 8/15/74)

9.	AFR 205-7	Communications Security (AFSC Supplement, 12/20/73 and ESD Supplement, 3/15/74)
10.	AFR 205-28	Communications Security for Nuclear Command and Control Communications
11.	AFR 300-8	Security Requirements for Automatic Data Processing Systems
12.	AFSCM 122-1	Nuclear Systems Safety Design Manual
13.	AFSCP 207-1	System Security Engineering
14.	DODD 5200.5	Communications Security
15.	DODD 5200.28	Security Requirements for Automatic Data Processing (ADP) Systems
16.	DODM 5200.28M	Automatic Data Processing Security Manual

2.6 GOVERNMENT DOCUMENTS PERTAINING TO SYSTEM ENGINEERING

1.	AFR 300-10	Computer Programming Languages
2.	DODD 5000.3	Test and Evaluation
3.	DODD 5000.31	Interim List of DOD Approved High Order Programming Languages (HOL)
4.	MIL-E 5400	Electronic Equipment, Airborne, General Specification for
5.	MIL-STD 499A	Engineering Management
6.	SAMSO Exhibit 73-3	Standard Engineering Practices for Computer Software Design and Development

3. GENERAL GUIDELINES

This section presents general guidelines that may be used by government or contractor software engineers for deriving and analyzing airborne software requirements that go into an RFP for full scale development of a weapon system or portion of a weapon system that includes airborne software. The software requirements should include all requirements that the airborne software must satisfy to perform its function adequately, including the system-level objectives that the airborne software must support. Each topic described in Sections 4 and 5 of this guidebook should be addressed for applicability to the airborne software requirements for the specific weapon system. As a minimum

- functional, performance, and interface requirements should be specified.
- design goals that are not requirements should be clearly marked as such.
- requirements should be specific and quantitative whenever possible, and they should be stated clearly, concisely, and unambiguously.
- each requirement should have a unique identifier (paragraph number).

3.1 SOFTWARE REQUIREMENTS ANALYSIS PROCEDURE

Software requirements analysis is a highly interactive process that requires considerable system and software engineering analysis and may require numerous iterations of tradeoffs to determine the best possible approach. Life cycle cost including subsystem supportability issues are also very much a part of the selection criteria for determining not only the types of computer equipment and computer programs but also the subsystem architecture and allocation of requirements to subsystem elements. The following discussion describes the type of procedural activity necessary to generate airborne software requirements for the RFP.

Figure 3-1 depicts the software requirements analysis procedure. Phase I of the procedure consists of studying the system-level requirements, the weapon system conceptual design, and the airborne software environment. As shown in Figure 1-2, the process begins after the final

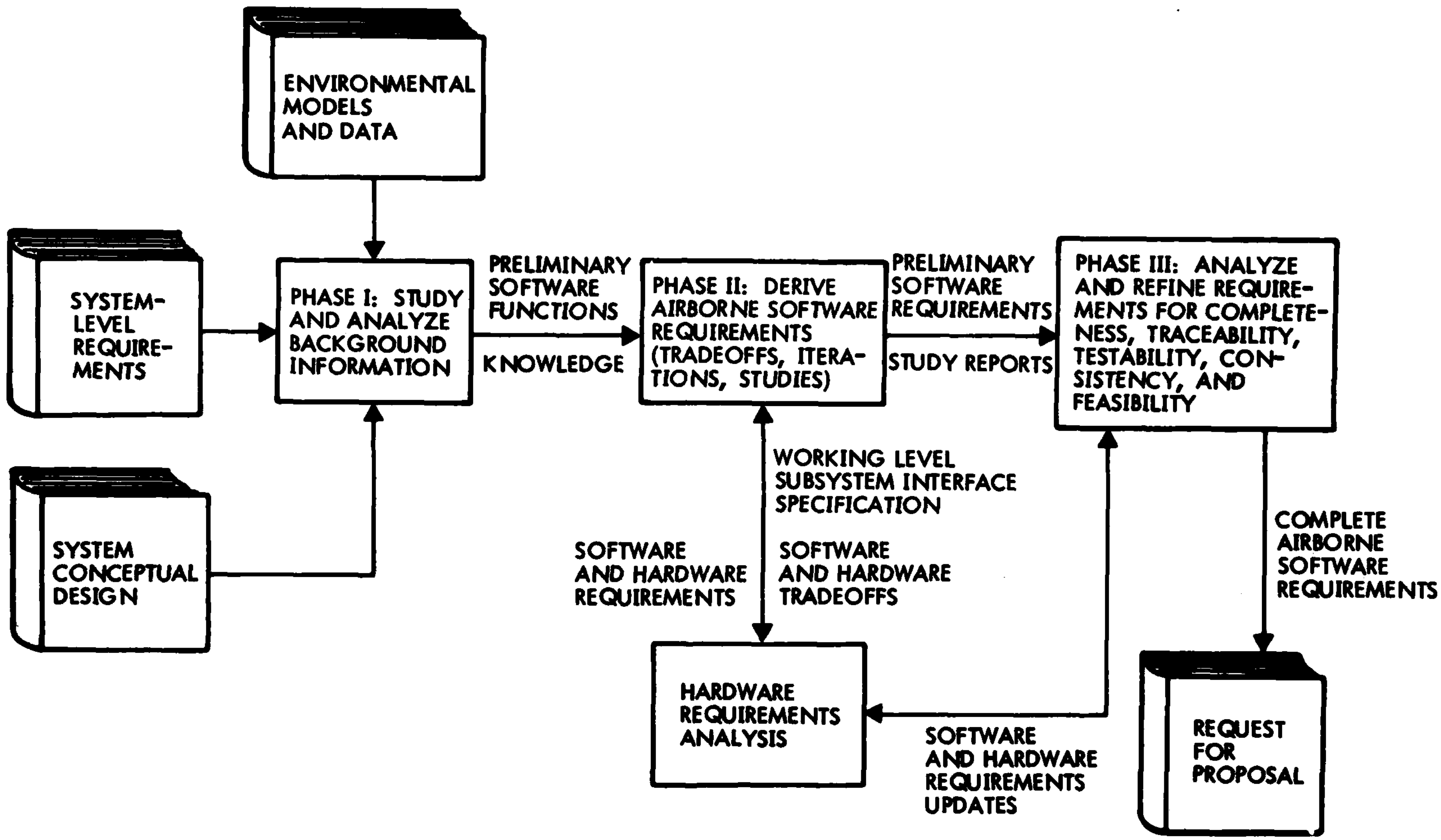


Figure 3-1. Airborne Software Requirements Generation Procedure

System Requirements Review (SRR), when the weapon system, conceptual design and preliminary system level requirements are available as a result of system requirements analysis. The first step in airborne software requirements analysis is to study and thoroughly understand the weapon system conceptual design and the system-level requirements and determine their impact on the airborne software requirements. This analysis yields a preliminary set of airborne software functions.

The environment in which the airborne software must operate must also be studied during Phase I. This includes the external environment of the airborne system, the aircraft, crew, and all other subsystems in the weapon system. The environment is described by mathematical models consisting of equations and values of parameters that appear in the equations. Models of the aerodynamics, atmospheric conditions, instrumentation performance, target characteristics, and spectrum of missions could, for example, be included in the environmental models.

At the completion of Phase I of the requirements analysis procedure, the software requirements analysts are prepared to begin deriving the detailed airborne software requirements as described in Sections 4 and 5 of this guidebook; this is Phase II of the procedure. During Phase II each type of requirement discussed in Sections 4 and 5 should be addressed. The FSED contractor may be required to derive some of the requirements in Section 5 after the contract is awarded. In order to produce requirements that correspond to acceptable system performance while minimizing weapon system life cycle cost, numerous tradeoffs, iterations, and studies must be performed during Phase II. Figure 1-2 shows hardware[†] requirements analysis taking place at the same time as software requirements analysis. The two analyses must be interactive so that the software requirements are compatible with the hardware requirements and to minimize weapon system life cycle cost. Hence, the hardware requirements and study results must be provided to the software requirements analysts, and the software requirements and study results must be provided to the hardware requirements analysts. The key document during this phase

[†] In this guidebook, the term "hardware" refers to digital processing equipment as well as all other non-software elements in the weapon system.

is a working level subsystem interface specification. The software requirements analysts must review the hardware outputs to determine any additional airborne software functions that are required by the hardware, to establish hardware/software interfaces, to confirm that the hardware will be adequate to support the airborne software functions, and to identify any constraints that the hardware imposes on the airborne software. The hardware requirements analysts should inform the software requirements analysts of any inconsistencies that they uncover between the software requirements and the hardware requirements.

Tradeoffs must be performed between hardware requirements and software requirements in order to minimize life cycle cost of the weapon system. For example, if the digital processing equipment is not required to have adequate memory for program and constant storage, then the cost of developing and maintaining the airborne software may be excessive compared with the cost of providing more memory. As another example, if on-board radar equipment has excessively high accuracy requirements, then the cost of the radar equipment may be higher than the cost of less accurate radar equipment plus the cost of requiring the airborne software to filter the radar measurements to yield the required weapon system accuracy.

At the end of Phase II of the requirements analysis, a baseline set of airborne software requirements will have been completed. The next phase of software requirements analysis is an assessment of those requirements relative to completeness, traceability, testability, consistency, and feasibility. These concepts are described in Sections 3.2 through 3.6 of this guidebook. This phase also includes factoring in updated hardware characteristics into the requirements. This third phase of software requirements analysis may result in many changes to the airborne software requirements; it is part of the iterative process in creating good requirements. This completes the software requirements analysis task. The resulting requirements can be incorporated into the Statement of Work portion of the FSED Request for Proposal, either directly or by reference. If the effort is being contracted, then provisions must be included in the validation phase contract for delivery of the data via CDRL as a study report, in addition to the preliminary version of the Computer Program Development Specification.

3.2 COMPLETENESS OF REQUIREMENTS

The airborne software requirements that go into the RFP should be as complete as possible. They should reflect all Air Force objectives and specify the relationship between the airborne software and the rest of the weapon system. Each topic discussed in Section 4 and applicable topics in Section 5 of this guidebook should be addressed. In addition, each system-level requirement and constraint should be analyzed to determine if it imposes any requirements on the airborne software. For example, if the system-level requirements include accuracy requirements, then the accuracy requirements must be allocated to the subsystems,[†] including the airborne software. The allocation of system-level accuracy requirements to the subsystems yields an error budget; this error budgeting activity is discussed in Section 4.3.

In addition to analysis of each system-level requirement and constraint for its impact on airborne software requirements, each subsystem that interfaces with the airborne software (receives data from or provides data to the airborne software) should be analyzed to determine what constraints and requirements it imposes on the airborne software (and what requirements the airborne software imposes on the subsystem). For example, the frequency at which data are needed from the airborne software by a subsystem in order for the subsystem to function properly would impose a processing (updating frequency) constraint on the airborne software. Similarly, the range of values and other characteristics of data needed from the airborne software by a subsystem would impose processing constraints on the airborne software. The update frequency and other attributes of data supplied to the airborne software by a subsystem may also impose processing requirements on the airborne software. For example, if the data received by the airborne software from a hardware item are known to be noisy and/or have compensable errors in it, this may impose a processing requirement on the airborne software to filter the data and/or compensate for the errors. The requirements may be in the form of airborne software output data accuracy requirements which

---

[†] In this guidebook, a subsystem is a hardware item, a software item, or a crew member that is an integral part (element) of the total weapon system.

are so stringent that the filtering and/or error compensation are necessary in order to satisfy the accuracy requirements. The statistical properties of the noise and compensable errors in the data received by the airborne software must also be supplied. The analysis of the interfacing subsystems to determine their effects on airborne software requirements results in a thorough understanding of the weapon system and the role of the airborne software in that system.

The goals to minimize life cycle cost and maximize reliability of the weapon system may not be explicitly stated in system-level requirements. Nevertheless, the impact of these goals on airborne software requirements must also be ascertained and taken into account in the requirements.

3.3 TRACEABILITY OF REQUIREMENTS

Each airborne software requirement in the RFP must be traceable to some underlying source, such as a system-level requirement; if this is not possible, then the requirement is not a real, defendable requirement. The source or basis of each requirement should be documented, as well as any analysis that leads from the basic source of the requirement to the requirement as it appears in the RFP. Documentation of the source of each requirement should take place at the time that the requirement is derived. This documentation does not necessarily have to be included in the RFP. The documentation is needed to record the requirements tracing information so that it can later be used in determining the effect on the airborne software requirements of a change in the weapon system hardware or a change in system-level requirements. The documentation also serves as proof that each requirement is traceable (real).

A requirement may trace back to a constraint imposed by an interfacing subsystem, the goal to minimizing the life cycle cost of the weapon system, system reliability considerations, or system-level requirements resulting from weapon system requirements analysis.

In addition to tracing each requirement back to its source, each requirement should have a separate paragraph number so that the airborne software design, code, and test plans can be precisely traced back to the requirements.

3.4 TESTABILITY OF REQUIREMENTS

All airborne software requirements in the RFP must be testable; i.e., an economically feasible method of testing each requirement should be identified during software requirements analysis. If a requirement has not been shown to be testable, then there is no assurance that the developing contractor will be capable of demonstrating that the airborne software end product satisfies that requirement. The methods of testing the requirements vary greatly with the types of requirements. Software testing methods are described in the SAE Guidebook for Verification, Validation, and Certification.

In order to be testable, requirements must be specific, unambiguous and quantitative whenever possible. Vague, general statements such as the following are not testable requirements:

- The airborne software shall function well under all operating conditions.
- The airborne software shall be developed in accordance with good development standards.
- The airborne software shall satisfy all system constraints.
- The airborne software shall provide features that facilitate future testing.
- The computer memory utilization by the airborne software shall be minimized in order to provide for future growth.
- The airborne software shall provide self tests for the airborne digital processors.
- The airborne software deliverables shall be appropriately identified.

Sections 4 and 5 of this guidebook describe how to produce testable requirements and include examples of testable requirements.

An economical method of testing each requirement should be documented by the software requirements analysts. The documentation of testing methods may be used by the developing contractor in estimating testing costs for his proposal and to aid him in preparing the test plans. The documentation also serves as proof that all the requirements are testable.

In order for the requirements to be testable, the environment in which the airborne software must operate should be defined. The requirements must define all the interfaces between the airborne software and other elements of the weapon system. All inputs must be identified, and the ranges of values, update frequencies, and other attributes of these inputs must be specified. In order to provide testability, the requirements must often include (either directly or by reference) mathematical models of the environment in which the airborne software must function. Both equations and parameter values should be included in model specifications. Statistical properties or ranges of values of perturbations in the model parameters should be given as well as nominal values of the model parameters. Functional models and error source models for each applicable subsystem that may influence the airborne software design and testing should be included or referenced. These could include system hardware performance characteristics, vehicle aerodynamics models, instrumentation characteristics, and descriptions of the environment external to the airborne system such as gravity models and atmospheric conditions. Error source models should describe the difference between actual subsystem characteristics and the ideal subsystem characteristics. For example, the statistical properties of radar measurement noise may be appropriate to include. The spectrum of missions over which the airborne system is expected to function must also be specified; e.g., the maximum time duration and range of a mission, altitude, speeds, and maneuvers anticipated during missions, and the characteristics of targets may be specified.

Certain airborne software performance characteristics may depend on values of airborne system parameters that are not fixed but are random variables or random processes that take on perturbed values. These system parameters may include radar measurement noise, atmosphere perturbations (winds), and mass properties uncertainties. As a result these airborne software performance characteristics are also random, and requirements on these performance characteristics must be statistical requirements. For example, a requirement that target miss must always be less than XYZ feet may be untestable for such a system. However, a requirement that the root mean squared value of target miss be less than

XYZ feet may be testable. In some cases, the statistical properties of airborne software performance characteristics can be computed exactly, e.g., by using linear statistics. In other cases, however, the statistical properties of the performance characteristics must be estimated empirically, e.g., using Monte Carlo statistical analysis. In the latter case, requirements on the statistical properties of the performance characteristics must include associated confidence levels less than 100 percent in order to make the requirements testable. Otherwise, an infinite number of Monte Carlo samples would be needed to prove that the requirements are satisfied. Alternatively, the requirements could specify precisely what tests are to be performed and what the results of the test must be; e.g., by providing a table of values of weapon system parameters for several cases, and requiring that the RMS of target miss be less than XYZ feet for those cases. This method of specifying performance requirements has the disadvantage that there is a danger that the developing contractor may design the airborne software to perform its best for those cases to be used in testing.

Deriving testable requirements that totally define the airborne software and cover all its capabilities, and demonstrating that they are testable requirements can be large, time consuming tasks. However, the benefits to be gained for the weapon system and the airborne software system, in particular, usually outweigh the expense involved, provided that the analysis is conducted early enough in the life cycle (e.g., validation phase) to effectively influence the system and software designs.

3.5 CONSISTENCY OF REQUIREMENTS

The airborne software requirements in the RFP must be self consistent; i.e., no requirements should conflict with any other requirements. For example, the required update frequency of some airborne software output data must be consistent with the frequency at which input data is updated. Consistency of the requirements should be checked by examining each requirement in relation to each of the other requirements for consistency and compatibility from a flow-oriented viewpoint and from a functional breakdown viewpoint. The requirements must also be checked for consistency with the airborne system-level requirements and with the constraints imposed by the interfacing subsystems.

3.6 FEASIBILITY OF REQUIREMENTS

It must be feasible to develop airborne software that will fulfill each requirement in the RFP. Requirements that have questionable feasibility, e.g., accuracy requirements, should be analyzed during software requirements analysis to prove their feasibility. One method of proving feasibility of a requirement is to produce a representative prototype airborne software design and demonstrate that it satisfies the requirement. For example, if an airborne navigation system has accuracy requirements imposed against it, a representative set of navigation equations could be designed and tested to show that the requirements are technically feasible. Any such prototype design and the analysis that lead to the design should be documented. This documentation should be provided to each potential bidding contractor to aid him in arriving at a technical approach for his proposal. Feasibility of each requirement should be assessed from the following standpoints: weapon system life cycle cost impact, computational capacity impact (throughput, memory requirements, wordlength, instruction set, etc.), technical feasibility, development schedule impact, and impact on interfacing subsystem requirements.

Feasibility of all the requirements when considered jointly must be assessed, as well as the feasibility of each requirement considered separately. In some cases each individual requirement may be feasible; however, certain requirements may be incompatible with other requirements. For example, the computer throughput limitations may be incompatible with the required update frequency of certain output data. Or the functional requirements or performance requirements may be inconsistent with the computer memory allocated for program and data storage. From these observations it is apparent that analyzing the requirements for feasibility overlaps analyzing the requirements for consistency.

3.7 REQUIREMENTS DOCUMENTATION

Whenever possible, the airborne software requirements documentation in the RFP should conform to the format for a CPCI Part I Development Specification as described in MIL-STD 483. This will simplify the developing contractor's task of converting the RFP requirements into CPCI development specifications. It will also simplify the Air Force procuring agency's task of reviewing the CPCI development specifications for consistency with the RFP requirements.

4. FUNCTIONAL AND PERFORMANCE REQUIREMENTS

This section describes how to derive and document airborne software functional, performance, and interface requirements that go into an RFP for Full Scale Engineering Development of a weapon system or portion of a weapon system that includes airborne software. All the requirements discussed in this section are critical airborne software requirements that should be addressed in the RFP. Each subsection describes one type of requirement, gives the purpose of the requirement, suggests how to derive the requirement and describes how to document the requirement in MIL-STD 483 format.

Many illustrative examples are included in this section which are based on a hypothetical weapon system referred to as F-X. The airborne software for this weapon system is identified as the F-X Airborne Software (F-X AS). The AS is used to perform navigation, visual display, and weapons delivery and control. The F-X AS consists of two CPCI's, the F-X Central Computer Software (F-X CCS) and the F-X Radar Data Processing Software (F-X RDPS). The F-X CCS performs the following functions:

- AS Master Executive
- Navigation
- Air-to-Air Support
- Air-to-Ground Support
- Self Test

The F-X RDPS performs the following functions:

- RDP Executive
- Antenna Control
- Search and Acquisition
- Tracking
- Displays Processing
- Radar Built-In Tests

The F-X CCS resides and executes in the F-X Central Computer and the F-X RDPS resides and executes in the Radar Digital Processor. The subsections that follow include illustrative examples that specify requirements for this hypothetical airborne software.

Table 4-1 summarizes the topics covered in this section and relates the topics covered with the applicable government documents (regulations, handbooks, and standards).

4.1 INTERFACE REQUIREMENTS

4.1.1 Purpose of Interface Requirements

Interface requirements in the RFP describe the functional relationship of the airborne software to other subsystems with which it must interface, i.e., subsystems which receive data from the airborne software or provide data to the airborne software. The interface requirements impose requirements and constraints, resulting from the interfaces, onto the airborne software. For example, an interface requirement on the airborne software to provide certain data to another subsystem imposes the requirement to produce that data, which may require processing by the airborne software. Attributes of that data that are specified in the interface requirements may impose constraints on the airborne software in producing the data. The interface requirements also impose requirements and constraints onto interfacing subsystems. The interface requirements are given in quantitative terms with tolerances where applicable. Interface requirements may specify the following characteristics (attributes and constraints) for each airborne software input/output data item:

- data item symbol
- definition
- units
- source subsystem (for airborne software input data)
- destination subsystems (for airborne software output data)
- source function (for airborne software output data)
- destination functions (for airborne software input data)

Table 4-1. Cross Reference for Section 4

Section Number	Subject	Applicable Government Documents
4.1	Interface Requirements	AFR 800-14 MIL-STD 483
4.2	Functional Requirements	AFR 800-14 MIL-STD 483
4.3	Performance Requirements	AFR 800-14 MIL-STD 483
4.4	Human Engineering Requirements	AFR 80-46 AFR 800-15 AFSC DH 1-3 MIL-H 46855A MIL-STD 483 MIL-STD 1472B
4.5	Safety Requirements	AFR 127-8 AFR 127-13 MIL-STD 454 MIL-STD 882A
4.6	Failure Compensation Requirements	MIL-STD 483
4.7	Self-Test Requirements	MIL-STD 483
4.8	Environment Requirements	MIL-STD 483
4.9	Data Base Requirements	MIL-STD 483

- paragraph number of source/destination function specification
- applicable airborne software mode of operation
- data rate
- message format
- maximum value
- minimum value
- precision
- means of data transfer between airborne software and interfacing subsystem
- memory storage locations
- sign/polarity conventions
- least significant bit/most significant bit conventions
- timing and sequencing constraints
- duration
- comments
- control

4.1.2 Derivation of Interface Requirements

The airborne software interface requirements are based on system-level requirements analysis and on analysis of the relationship between the airborne software and interfacing subsystems. Each interfacing subsystem must be analyzed to determine what data it needs from the airborne software, what data it provides to the airborne software, and the constraints and characteristics of those interface data. Each function of the airborne software must also be analyzed to determine what data the function needs from interfacing subsystems, what data the function provides to the interfacing subsystems, and the constraints and characteristics of those interface data.

4.1.3 Specification of Interface Requirements

If the airborne software requirements in the RFP are documented in MIL-STD 483 format, then the interface requirements should appear in Paragraphs 3.1.1, 3.1.1.1, and 3.1.1.2 of the document.† Paragraph 3.1.1 contains only general introductory information, no quantitative information. Paragraph 3.1.1.1 contains an interface block diagram showing the functional relationship between the airborne software and interfacing subsystems. Paragraph 3.1.1.2 and its subparagraphs contain the detailed interface requirements in quantitative terms, including the characteristics of the interface data. The requirements for the interface between the airborne software and each interfacing subsystem are presented in a separate subparagraph of Paragraph 3.1.1.2. Each subparagraph includes a table of inputs to the airborne software from the interfacing subsystem and a table of outputs from the airborne software to the interfacing subsystem. Each table has the interfacing data running vertically and the interfacing data characteristics running horizontally. Each subparagraph of Paragraph 3.1.1.2 also includes interface constraints and requirements that do not appear in the tables.

The following pages are examples of interface requirements for the airborne software CPCI designated F-X CCS in the hypothetical F-X weapon system.‡

---

†When interfaces are complex or involve subcontractors, separate government agencies, etc., they can be documented in a separate document such as an Interface Control Document (ICD) that is referenced in the airborne software requirements documentation.

‡Note that an equivalent analysis should be included for each CPCI, e.g., the F-X RDPS. If CPCI's have not been identified, the airborne software system can be treated as a whole.

Sample

3.1.1 Interface Requirements

This paragraph specifies requirements imposed on the F-X CCS because of its relationship to other equipment and software. This includes the input/output requirements that exist between the F-X CCS and the interfacing systems, and it includes the constraints which ensure that these interface requirements are satisfied. The interfacing systems are:

1. *Central Computer, CC*
2. *Central Computer Assembler, CCA*
3. *Multiplex (MUX) buses*
4. *Heads-Up Display, HUD*
5. *Navigation Control Indicator, NCI*
6. *Multi-Purpose Indicator, MPI*
7. *Horizontal Situation Indicator, HSI*
8. *Analog Radar Signal Processor, ARSP*
9. *Attitude Heading Reference Set, AHRS*
10. *Life Data Recorder, LDR*
11. *Radar Data Processing Software*
12. *Inertial Navigation System, INS*
13. *Lead Computing Gyro, LCG*
14. *Armament Control Set, ACS*
15. *Built In Test (BIT) Control Panel, BITCP*
16. *Avionic Status Panel, ASP*

3.1.1.1 Interface Block Diagram

The relationship of the F-X CCS to its interfacing elements is shown in Figure 4-1. Each subparagraph number appearing in the figure refers to a paragraph which describes in detail the relationship of that particular system element with the F-X CCS. The configuration item number, CI-XXXXX, is included for each interfacing element, where applicable.

3.1.1.2 Detailed Interface Definition

This paragraph specifies the relationship between the F-X CCS and interfacing subsystems, and it specifies design requirements imposed upon other equipment and computer programs as a result of the requirements on this CPCI. This information is given in quantitative terms with the range of acceptable values where applicable and to the level of detail necessary to permit design of the CPCI. The interface requirements are delineated in the following subparagraphs.

3.1.1.2.1 Central Computer

3.1.1.2.1.1 Inputs

This section is not applicable to this specification.

3.1.1.2.1.2 Outputs

This section is not applicable to this specification.

3.1.1.2.1.3 Constraints

The F-X CCS shall use CC locations 123456_8 through 712345_8 The CC is described in the F-X Central Computer Programming Reference Document TI-234.

3.1.1.2.2 Central Computer Assembler

3.1.1.2.2.1 Inputs

This paragraph is not applicable to this interface.

3.1.1.2.2.2 Outputs

This paragraph is not applicable to this interface.

Sample

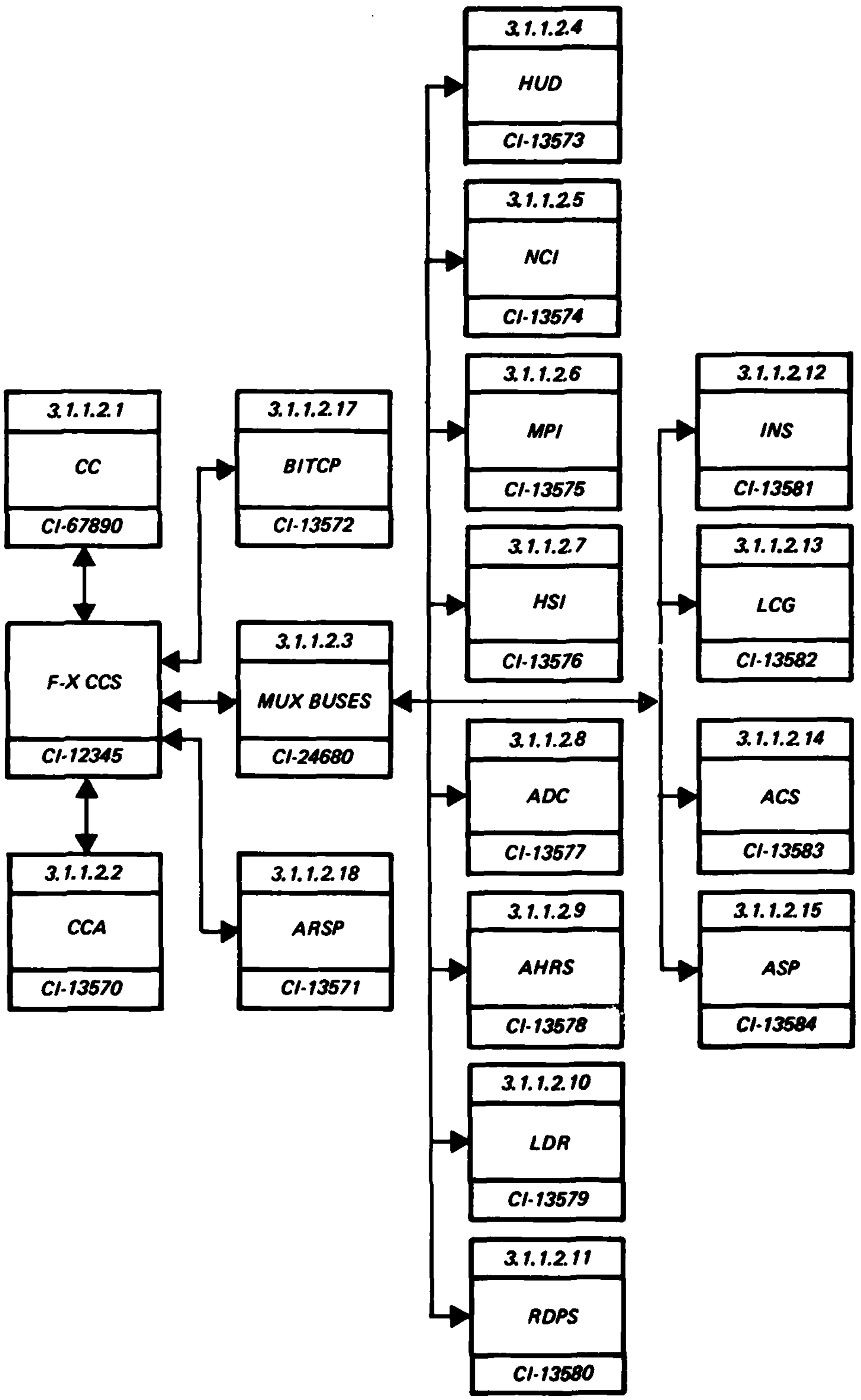


Figure 4-1. Interface Block Diagram

Sample

3.1.1.2.2.3 Constraints

The F-X CCS shall be coded in JOVIAL J73/I as defined by MIL-STD 1589 and in F-X central computer assembly language as defined in the F-X Central Computer Master Programming Reference Manual TI-234.

3.1.1.2.3 Multiplex Buses

3.1.1.2.3.1 Inputs

The F-X CCS shall accept inputs from the multiplex data buses as shown in Table 4-2.

3.1.1.2.3.2 Outputs

The F-X CCS shall provide outputs to the multiplex data buses as shown in Table 4-3.

3.1.1.2.3.3 Constraints

The multiplexing restrictions for the F-X multiplex data buses shall be in accordance with the F-X Multiplex Data Bus Reference Document TI-345.

3.1.1.2.4 Heads-Up Display

3.1.1.2.4.1 Inputs

The F-X CCS shall accept inputs from the heads-up Display (HUD) as specified in Table 4-4.

3.1.1.2.4.2 Outputs

The F-X CCS shall provide outputs to the heads-up display as shown in Table 4-5.

3.1.1.2.4.3 Constraints

The F-X CCS interface with the HUD shall be in accordance with the following constraints:

a. *Symbol positioning polarities and coordinates shall be in accordance with the HUD Reference Manual TI-456, Volume II, paragraph 3.0.*

•
•
•

Sample

Table 4-2. Inputs to F-X CCS from Multiplex Data Bus

Symbol	Name	AS Mode	Sample Frequency 1/sec	Bus Number	Time Slot	Destination	
						Function	Paragraph
θ_D	Depression Angle	Air-to-Air	5	1, 3	18	Air-to-Air	3.2.1
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮

Table 4-3. F-X CCS Outputs to Multiplex Data Bus

Symbol	Name	AS Mode	Sample Frequency 1/sec	Bus Number	Time Slot	Source	
						Function	Paragraph
S_p	Symbol Position	Air-to-Air	5	2, 4	23	Air-to-Air	3.2.1
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮

Table 4-4. Inputs to F-X CCS from HUD

Symbol	Name	Units	Limits		Precision	Frequency 1/sec	Destination	
			Minimum	Maximum			Function	Paragraph
S_R	Symbol Reject	NA	0	1	0.5	5	Air-to-Air	3.2.1
θ_D	Depression Angle	deg	0	360	0.01	5	Air-to-Air	3.2.1
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮

Table 4-5. F-X CCS Outputs to HUD

Symbol	Name	Units	Limits		Precision	Frequency 1/sec	Source	
			Minimum	Maximum			Function	Paragraph
S_{CS}	Status Control Signals	NA	0	1	0.5	5	Air-to-Air	3.2.1
S_p	Symbol Position	deg	0	360	0.01	5	Air-to-Air	3.2.1
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮

4.2 FUNCTIONAL REQUIREMENTS

4.2.1 Purpose of Functional Requirements

Functional requirements in the RFP for Full Scale Engineering Development specify the processing performed by the airborne software. An airborne software function is a data processing operation to which quantitative performance requirements can be meaningfully applied. It is the lowest level of airborne software breakdown appearing in the RFP. A function may include decision logic as well as arithmetical processing.

The functional requirements describe the inputs, outputs, and processing performed by each function, and they specify the quantitative performance requirements of each function when applicable.

4.2.2 Derivation of Functional Requirements

Derivation of functional requirements begins with establishing and defining all the functions that must be performed by the airborne software. Some functions may be identified in the system-level requirements. The remaining functions must be identified during software requirements analysis as a result of studying the weapon system; e.g., some of the remaining functions are based on the requirements imposed by subsystems which interface with the airborne software. The combination of all airborne software functions should constitute all desired processing capabilities of the airborne software; i.e., every processing operation performed by the airborne software should fall under one of the functions.

After all the functions are identified and defined, the required outputs from each function should be established and described. This includes determining the destinations, units of measure, limits/ranges of values, accuracy/precision, and frequency of update of each output of a function. Some of these characteristics of the output data are based on requirements of the subsystems and functions that interface with the function, and some of the characteristics are based on human engineering considerations and weapon system-level requirements.

After the required outputs to each function are defined, the processing performed by each function must be determined. This task can be accomplished by first establishing the precise objectives of each function

and then analyzing what processing is required to accomplish those objectives. In some cases, the required processing is straightforward, and mathematical equations and logic that will accomplish the processing objectives can be easily derived. In other cases, however, the required processing is not so straightforward; e.g., when there are many mathematical algorithms that could accomplish the processing objectives and when compromises and tradeoffs must be made in the algorithm design. In these cases the processing requirements become the processing objectives together with functional performance requirements that specify quantitatively how well those objectives must be met. The performance requirements should include tolerances and quantitative constraints whenever applicable, e.g., limits on acceptable degraded performance. In some cases, the performance requirements are best expressed in statistical terms; e.g., the single shot kill probability shall be at least X percent, or the standard deviation of the output error shall be no greater than Y units. These quantitative performance requirements are generally based on weapon system-level performance requirements or requirements imposed by other subsystems onto the airborne software. The derivations of airborne software performance requirements are described in Section 4.3, below. The processing requirements should include relative sequencing, periodicities, options, and other important relationships of each function as appropriate.

After the processing requirements of a function have been determined, then the required inputs to the function can be identified and described, including the sources of the input data, units of measure, limits/ranges of values, precision, and frequency of arrival of the input data.

4.2.3 Specification of Functional Requirements

If the requirements are documented in MIL-STD 483 format, then the functional requirements should appear in Paragraph 3.2 and its subparagraphs. Paragraph 3.2 should contain general and descriptive material about the functional requirements. It should include a functional block diagram or the equivalent illustrating the functional operation of the software, the relationships between the functions, and the relationships between software functions and the interfacing subsystem functions.

The detailed requirements for each software function should be presented in a separate subparagraph of Paragraph 3.2. Each subparagraph should begin with a narrative description of the subject function and its relation to other functions and subsystems.

Next the subparagraph should describe the required inputs to the function, usually in a table of input variables versus their characteristics (definition, sources, units of measure, limits/ranges of values, accuracy/precision required, frequencies of arrival, etc.).

Then the subparagraph should present the processing requirements of the function. It should describe the exact intent of processing operations, and it should include textual descriptions and symbol definitions for all mathematical equations and logic included. It should also specify required accuracies of processing, sequences and timing of processing operations, and relevant restrictions and limitations on the processing required. It should include quantitative performance requirements where appropriate.

Finally, the subparagraph should describe the required outputs from the function, usually in a table of output variables versus their characteristics (definitions, destinations, units of measure, accuracy/precision required, frequency of update, etc.).

The following pages are examples of functional requirements for the hypothetical F-X CCS CPCI.[†]

[†]Note that an equivalent analysis should be performed for each CPCI, e.g., the F-X RDPS.

Sample

3.2 DETAILED FUNCTIONAL REQUIREMENTS

The F-X CCS performs the following functions:

1. Executive
2. Navigation
3. Air-to-Air Delivery Support
4. Air-to-Ground Delivery Support
5. Self-Test

The detailed requirements for each of these functions are specified in the following paragraphs, and the relationship between functions is illustrated by the functional block diagram, Figure 4-2.

3.2.1 Executive

This function imposes order and structure to the F-X CCS. The function provides system initialization upon start-up or after restart from a power interrupt. This function also schedules the order and rate of execution and I/O operation for each functional module. This function provides facilities for servicing all interrupts, external and internal.

3.2.1.1 Inputs

This function accepts the inputs as delineated in Table 4-6.

3.2.1.2 Processing

This function of the F-X CCS shall execute the necessary computations at three basic rates: minor cycle (20 per second), intermediate cycle (10 per second), and major cycle (5 per second). The F-X CCS shall generate the outputs as specified in Paragraph 3.2.1.3 using the input data specified in Paragraph 3.2.1.1 and the processed data specifications of Table 4-7.

3.2.1.2.1 F-X CCS Initialization

The executive function shall initialize the F-X CCS upon start-up or after restart from a power interrupt. This consists of setting numerous executive variables and flags to their appropriate initial values and commanding the functions to perform their initialization computations, as shown in Figure .... .

3.2.1.2.2 Scheduling Execution of Each Function

The execution function shall schedule the execution of each F-X CCS function based on the mode of operation and the required execution frequency. The scheduling is based on real-time-interrupt discretes received from the central computer clock. ...

3.2.1.3 Outputs

The executive function shall output data derived from the processing described in Paragraph 3.2.1.2 and its subparagraphs, as specified in Table 4-8.

Sample

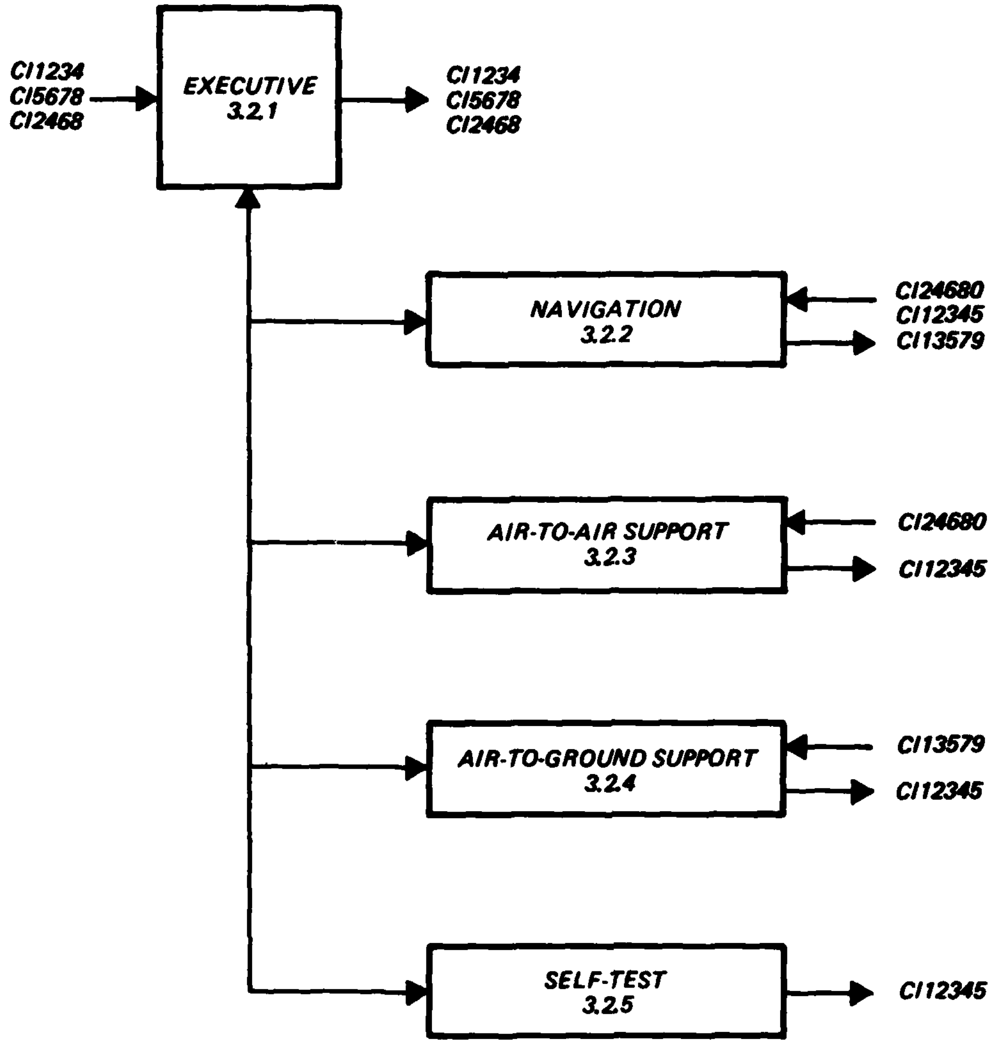


Figure 4-2. F-X CCS Functional Block Diagram

Sample

Table 4-6. Executive Input Data

Constant or Symbol	Name	Source (Paragraph)
t_R	Reference time	3.1.1.2.1.1
i_{RTI}	Real time interrupt	3.1.1.2.1.1
ALT	Altitude above target	3.2.2.1.3
⋮	⋮	⋮

Table 4-7. Executive Processed Data

Symbol	Name	Units	Value	
			Maximum Absolute	Least Significant
t_{LI}	Time of last interrupt	sec	20,000	0.0005
Δt_{LI}	Elapsed time since last interrupt	sec	1.0	0.0001
⋮	⋮	⋮	⋮	⋮

Table 4-8. Executive Output Data

Constant	Name	Source	Value		Units of Measure	Destination	
			Maximum Absolute	Least Significant		Function	Paragraph
t_{NAV}	Navigation update time	3.2.1.2.2	20,000	0.0005	sec	Navigation	3.2.2.1
i_{ST}	Self-test discrete	3.2.1.2.9	ON	OFF	Discrete	Self-test	3.2.8.1
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮

4.3 PERFORMANCE REQUIREMENTS

4.3.1 Purpose of Performance Requirements

Airborne software performance requirements in an RFP for Full Scale Engineering Development specify in quantitative terms how well the airborne software must satisfy its processing objectives. A performance requirement may be directed at a single function or it can be directed at the airborne software as a whole. A performance requirement may be based on system-level performance requirements such as accuracy requirements or it may be based on requirements imposed onto the airborne software by other subsystems that interface with the airborne software. A performance requirement is generally used when approximations, compromises, and tradeoffs must be made in the design in order to satisfy a processing objective within the system constraints, e.g., within computer storage and throughput limitations.

4.3.2 Derivation of Performance Requirements

It is generally necessary to transform weapon system-level performance requirements into performance requirements imposed on the airborne software; this task is part of software requirements analysis. First each system-level performance requirement must be analyzed to determine if the airborne software performance could affect the satisfaction of that requirement. If the airborne software performance could affect the satisfaction of a system-level performance requirement, then all other subsystems which could also affect the satisfaction of the requirement should be identified. The airborne software performance requirement that reflects the system-level performance requirement must be designed so that if the airborne software and all other subsystems satisfy their performance requirements, then the system-level performance requirement will be met. Therefore, the derivation of the airborne software performance requirement must be coordinated with requirements generation for all other subsystems that may affect satisfaction of the system-level requirement. In addition, the performance requirements on the subsystems should not be more stringent than necessary to ensure that the system-level performance requirements will be met; otherwise, the cost of the subsystems may be higher than necessary. For example, suppose a system-level performance requirement is that the standard deviation (σ) of crossrange target miss for a

bomb must not exceed σ_M feet; suppose the target miss is due to airborne software approximations, human errors and instrument errors; and suppose the three contributions to crossrange miss are uncorrelated. Then the required standard deviations of the contributions to crossrange miss due to the three contributors σ_{AS}, σ_{HMN}, and σ_{INST} must satisfy

$$\sigma_{AS}^2 + \sigma_{HMN}^2 + \sigma_{INST}^2 = \sigma_M^2$$

in order to ensure that the system-level requirement will be met and that the subsystem-level requirements are not more stringent than necessary. The budgeting of the system-level performance requirement between the subsystems (e.g., the selection of σ_{AS}, σ_{HMN}, and σ_{INST} which satisfy the above equation) should be accomplished with a goal of minimizing weapon system life cycle cost.

As the above example suggests, a system-level performance requirement influences the method of specifying airborne software performance requirements as well as providing the quantitative data on which the airborne software requirement is based. In the above example, the simplest way to specify the airborne software accuracy requirement on crossrange miss is to specify a standard deviation σ_{AS} of crossrange miss contribution due to the airborne software.

Part of software requirements analysis is to demonstrate that if the airborne software and all other subsystems satisfy their performance requirements, then the system-level performance requirements will be met; i.e., show that all the performance requirements are consistent.

An important consideration in deriving performance requirements is testability of the requirements. The requirements must be stated so that the airborne software can be tested against its requirements by economically feasible methods. For example, an absolute tolerance on the error in some airborne software output parameter may be impossible to test because of the large range of values and the many possible combinations of values of the airborne software input data; it may not be economically feasible to prove that the airborne software error can never exceed the tolerance. In this case, it is necessary to state the accuracy requirement as a statistical requirement. For example, require that the probability of

the error exceeding the tolerance must be no more than 10 percent. This approach to writing performance requirements requires that statistical properties of the relevant input data are known or can be determined. Otherwise, the statistical properties of the airborne software performance could not be determined during testing. Information that can be used to determine the statistical properties of the airborne software input data should be included in the requirements, either directly or by reference. This information may be in the form of statistical properties of weapon system parameters and error sources.

For a complex weapon system, it may be impossible to analytically determine the exact statistical properties of airborne software performance, and it may be necessary to use Monte Carlo analysis to estimate the statistical properties of performance. For this airborne system the statistical performance requirements must include confidence levels in order to make the performance requirements testable by an economically feasible number of Monte Carlo samples. For example, require that the probability of an airborne software error exceeding a tolerance must be no more than 10 percent with at least 80 percent confidence. Or require that the standard deviation of an error parameter be no greater than σ_{AS} with 90 percent confidence.

Confirmation that performance requirements are feasible (realizable) is also part of software requirements analysis. This task may require development of prototype (representative) designs of critical portions of the airborne software and testing the prototype designs to verify that they satisfy the performance requirements.

The procedure for deriving airborne software performance requirements based on requirements imposed on the airborne software by other subsystems is analogous to the system-level procedure described above.

Budgeting a system-level accuracy requirement to the subsystems is more difficult when the system-level accuracy requirement is specified in terms of a required Probability of Kill (PK) or the Circular Error Probability (CEP). If CEP is specified then in order to specify the airborne software accuracy requirement, it is necessary to know the joint probability distribution of the contributions to miss due to all other

subsystems. If PK is specified, then it is also necessary to know the probability distribution of the effective kill radius of the exploding device. This information is generally not available during software requirements analysis. For this reason, it is desirable to re-specify the system-level accuracy requirements in terms of tolerances on the root summed squared value of mean values of the three components of miss and on the root sumed squared value of the standard deviations of the three components of miss; i.e., the mean values u_x, u_y, and u_z of the three components of miss must satisfy

$$\sqrt{u_X^2 + u_Y^2 + u_Z^2} \leq u_{MAX}$$

and standard deviations σ_X, σ_Y, and σ_Z of the three components of miss must satisfy

$$\sqrt{\sigma_X^2 + \sigma_Y^2 + \sigma_Z^2} \leq \sigma_{MAX}$$

If the three components of miss are assumed to be Gaussian (normal) random variables, then the tolerances u_{MAX} and σ_{MAX} can be selected so that if the above tolerances are satisfied, then the probability of kill or CEP requirement will be satisfied. (The central limit theorem of statistics states that if the total miss is a summation of uncorrelated miss contributions, then the total miss tends to be a Gaussian (normal) random variable, even if the miss contributions are not Gaussian.)

If the contributions to miss due to the subsystems are statistically uncorrelated,† each subsystem should have the following accuracy requirement imposed against it:

$$\sqrt{\sigma_{X_i}^2 + \sigma_{Y_i}^2 + \sigma_{Z_i}^2} \leq \sigma_{MAX_i}$$

where σ_{X_i}, σ_{Y_i}, and σ_{Z_i} are the standard deviation of the three components of the contribution of miss due to the i^{th} subsystem. To ensure that the

†If the contributions to miss due to any two subsystems are correlated, then these two subsystems can be combined into a single subsystem for the purpose of accuracy specification.

probability of kill or CEP requirement will be met, the subsystem tolerances must satisfy

$$\sum_i \sigma_{MAX_i}^{\;2} = \sigma_{MAX}^{\;2}$$

Generally, one subsystem, e.g., the j^{th} subsystem, is responsible for compensating for any bias in the system, i.e., responsible for minimizing the mean miss. This subsystem should have the following additional accuracy requirement imposed against it:

$$\sqrt{u_X^{\;2} + u_Y^{\;2} + u_Z^{\;2}} \leq u_{MAX}$$

where u_X, u_Y, and u_Z are the mean values of the three components of the total miss.

If only two components of miss (X and Y) are defined for a given system, then u_Z, σ_Z, and σ_{Z_i} can be deleted from the above expressions.

In order to make them economically testable, it may be necessary to include confidence levels less than 100 percent in the above accuracy requirements.

4.3.3 Specification of Performance Requirements

If the airborne software requirements are documented in MIL-STD 483 format, then a performance requirement that pertains to only one function should be included in the functional requirements (processing requirements) for that function, i.e., in the appropriate subparagraph of Paragraph 3.2. Functional requirements that cannot be assigned to a single function but pertain to the CPCI as a whole should be placed in the special requirements subparagraph of Paragraph 3.2. The special requirements subparagraph follows immediately after the functional requirements subparagraphs. In any case, functional requirements should be stated clearly and concisely.

4.4 HUMAN ENGINEERING REQUIREMENTS

4.4.1 Purpose of Human Engineering Requirements

Human engineering requirements specify the interface requirements between the airborne software and the crew, and they characterize (model) the performance of the crew members as subsystems in the weapon system. The human engineering interface requirements may specify maximum allowed time for airborne software responses to input data, maximum display densities of information, clarity requirements on data presented in displays, and other attributes of data provided to the crew by the airborne software. Models of the crew may specify minimum times for human decision-making and for response to stimuli, they may describe the accuracy of human performance, and they may describe the human decision-making logic. The human engineering requirements should provide enough information about the crew performance so that the crew members can be treated as subsystems in the airborne system during airborne software design.

4.4.2 Derivation of Human Engineering Requirements

Some human engineering requirements are based on system-level human performance and human engineering requirements. Others are based on measuring human performance and reaction times under simulated in-flight conditions. The government documents listed in Section 2.3 and the SAE Guidebook for Software Quality Assurance may be helpful in deriving human engineering requirements.

4.4.3 Specification of Human Engineering Requirements

If the airborne software requirements are documented in MIL-STD 483 format, then the human engineering requirements should appear in the first subparagraph of the last functional requirements paragraph (Special Requirements); the title of this subparagraph should be "Human Performance." The subparagraph should cite appropriate paragraphs of the system/system segment specification which establish human performance and human engineering requirements. The subparagraph should be similar to an interface requirements paragraph; it should list all the data provided to the crew by the airborne software and all the data provided to the airborne software by the crew, along with attributes of these data. The

subparagraph should also include any constraints associated with interface between the airborne software and crew, and it should provide models of the relevant performance characteristics of the crew.

4.5 SAFETY REQUIREMENTS

4.5.1 Purpose of Safety Requirements

Airborne software safety requirements in an RFP for Full Scale Engineering Development relate to providing safety to the crew, other personnel, allied forces, and population and structures other than military targets. Safety requirements may specify airborne software features for detecting subsystem malfunctions and dangerous situations, providing warning signals to the crew in dangerous situations, implementing back-up modes of operation when a subsystem fails, and initiating corrective action in a dangerous situation.

4.5.2 Derivation of Safety Requirements

Safety requirements depend on the weapon system and its mission. An analysis of the subsystems in the airborne system may reveal critical elements that could fail and cause a safety hazard. The airborne software may be required to detect such a failure, provide a warning signal to the crew when the failure occurs, and compensate for the failure. Analysis of the missions performed by the airborne system and its operating environment may identify potentially hazardous situations that could occur during a mission. The airborne software may be required to detect these hazardous situations, provide a warning to the crew when a hazardous situation occurs, and initiate corrective action to compensate for or eliminate the hazardous situation. Flight safety requirements are generally based on system-level requirements derived from system-level requirements analysis. The government documents listed in Section 2.4, above, may also be useful in deriving flight safety requirements.

4.5.3 Specification of Safety Requirements

If the airborne software requirements are documented in MIL-STD 483 format, then a function providing safety should be documented as a functional requirement in a subparagraph to Paragraph 3.2 (Detailed Functional Requirements). Specification of functional requirements is

covered in Section 4.2 of this guidebook. General safety requirements that do not pertain to any one function but pertain to the CPCI or airborne software as a whole should be documented in last functional requirements subparagraph (Special Requirements) of Paragraph 3.2.

4.6 FAILURE DETECTION AND COMPENSATION REQUIREMENTS

4.6.1 Purpose of Failure Detection and Compensation Requirements

Failure detection and compensation requirements specify airborne software functions and performance characteristics that will increase the reliability of the weapon system.† The airborne software may be required to detect and compensate for subsystem failures and for off-nominal system parameters that would otherwise cause failure of a mission. The requirements increase the probability of successfully completing a mission, and they reduce system maintenance costs and time.

4.6.2 Derivation of Failure Detection and Compensation Requirements

Failure detection and compensation requirements are based on analysis of other subsystems in the airborne system. A subsystem may have a relatively short mean time between failures or it may contain error sources that could reduce the probability of successfully completing a mission. If software requirements analysis demonstrate that the airborne software receives sufficient information to detect and compensate for the subsystem failures and/or to compensate for the error sources in the system, then it can be required to perform this function. Failure detection and compensation requirements such as these may also be based on the system-level requirements. In arriving at a failure detection and compensation requirement, a life cycle cost tradeoff analysis should be performed to weigh the cost of including this function in the airborne software with the cost of building and operating additional airborne systems and increased maintenance of those systems to provide the same total weapon system reliability and effectiveness.

†Reliability of the airborne software itself (i.e., relative to errors in the airborne software) is addressed in the quality assurance provisions (Section 4) of the requirements. The SAE Guidebook for Quality Assurance and the SAE Guidebook for Verification, Validation, and Certification cover quality assurance and testing methods required to assure reliable software.

4.6.3 Specification of Failure Detection and Compensation Requirements

If the requirements are documented in MIL-STD 483 format, then failure detection and compensation requirements specify additional airborne software functions that will increase reliability of the weapon system, and these functions should be specified in subparagraphs of Paragraph 3.2 (Detailed Functional Requirements) of the requirements. Section 4.2 of this guidebook describes how to specify functional requirements.

4.7 SELF-TEST REQUIREMENTS

4.7.1 Purpose of Self-Test Requirements

Self-test requirements are a special class of failure detection requirements; they specify that the airborne software periodically test the airborne digital processors. The requirements may specify that if a processor malfunction is detected, then a warning is provided to the crew and a backup system is automatically put into service.

4.7.2 Derivation of Self-Test Requirements

Self-test requirements are based on analysis of the digital processors in the airborne digital processor subsystem and analysis of processor status monitoring devices. Instruction test routines are generally required to execute each digital processor through a sequence of typical computations and compare the results (bit-for-bit) with pre-stored answers. Any disagreement with the pre-stored answers indicates a processor malfunction. Another test that may be required is a memory checksum test to ensure that data stored in a computer memory did not change value unintentionally. If the airborne digital processor includes redundant processors, i.e., identical processors that perform exactly the same computations, then the results of the computations can be compared to identify a processor malfunction. If processors are triply redundant, i.e., three identical processors, then a malfunction can also be isolated to one of the three processors, and the two remaining good processors can continue to perform the processing. With doubly redundant processors, reasonableness checks can sometimes be used to determine which processor has malfunctioned. Status monitoring devices (which measure parameters

such as processor temperature, voltage, and current, input/output rates, and timing of routines) can also aid in identifying and isolating processor malfunctions.

The need for self-test requirements also depends on the reliability of the computers (mean time between failures) and on the possible losses if a malfunction goes undetected and/or uncompensated. The life-cycle-cost (acquisition and maintenance) of self-test equipment and software should be compared with the dollar cost penalty of undetected and uncompensated processor failures, weighted by the probability of a processor failure during a mission.

4.7.3 Specification of Self-Test Requirements

Self-tests can be treated as airborne software functions. If the requirements are documented in MIL-STD 483 format, the requirements for those self-test functions should be specified in subparagraphs of Paragraph 3.2 (Detailed Functional Requirements).

4.8 ENVIRONMENT REQUIREMENTS

4.8.1 Purpose of Environment Requirements

Environment requirements specify the environmental conditions in which the airborne software must function and satisfy its performance requirements. The requirements describe the environmental parameters and mathematical models (equations and coefficients) that affect the design and performance of the airborne software. Whereas interface requirements discussed in Section 4.1, above, specify the characteristics of data that is communicated between the airborne software and interfacing subsystems, environment requirements describe the performance of the interfacing subsystems and of subsystems that do not interface directly with the airborne software. The subsystems that are described are those whose performance could affect the satisfaction of airborne software performance requirements discussed in Section 4.3, above. Each subsystem is described adequately to determine the effect of that subsystem performance on airborne software performance. The descriptions of the environmental parameters should specify the anticipated characteristics of the parameters for all airborne software installations and under all

mission conditions. Nominal values and ranges or statistical properties of perturbations relative to the nominal values should be specified. The environmental parameters may include mission parameters such as mission durations, ranges, altitudes, velocities, maneuvers, and target characteristics. The environmental parameters may also include airborne system parameters, such as aerodynamic properties, mass properties, propulsion parameters, and performance characteristics of subsystems in the airborne system. Finally, the environmental parameters may include external environmental parameters such as gravity models, atmospheric conditions (wind, visibility, radar transmission, etc.), and effects of enemy countermeasures (electronic countermeasures, nuclear radiation, anti-aircraft fire, surface-to-air missiles, etc.).

In order to precisely specify the environment (unambiguously) it is often necessary to include equations that model the environment and show how the environmental parameters relate to the airborne software functions and the physical laws that govern the subsystem performance. For example, if aerodynamic parameters are included, the equations that relate these parameters to the forces and moments acting on the aircraft should be included, and if the instrument error parameters are described, the equations that relate the error parameters with the measurements received from those instruments should be specified.

Mathematical models that specify the relationship between the airborne software outputs and the performance parameters (measures of airborne software performance that have requirements imposed against them) are also included in the environment requirements. For example, if the airborne software requirements include accuracy requirements on target miss, then equations and coefficients that determine target miss resulting from airborne software outputs should be included in the environment requirements.

4.8.2 Derivation of Environment Requirements

Environment requirements are derived by analyzing each subsystem that interfaces with the airborne software. Those performance characteristics and mathematical models of each subsystem that may affect the design or performance of the airborne software should be included in the

environment requirements. In addition, all external environmental parameters and characteristics that affect the inputs to the airborne software should be identified, analyzed, modeled, and included in the environment requirements. For example, if a radar subsystem interfaces with the airborne software, the atmospheric conditions may affect the characteristics (noise) of radar data that is provided to the airborne software. Hence, the anticipated atmospheric conditions should be analyzed, modeled, and included in the environment requirements. As another example, target characteristics and evasive maneuvers that are performed by a target will affect measurements of the target location that are input to the airborne software. Therefore, anticipated target characteristics and evasive maneuvers should be analyzed, modeled, and included in the environmental requirements.

Mathematical models that relate the airborne software outputs with the airborne software performance parameters are derived by analyzing the relationship between the airborne software, the weapon system and its external environment. This analysis should result in equations and associated coefficients that determine the value of the airborne software performance parameters based on a given set of airborne software outputs.

Whenever coordinate systems are included in environmental models, the coordinate axes should be defined.

4.8.3 Specification of Environment Requirements

If the software requirements are documented in MIL-STD 483 format, then the environmental requirements are presented in Paragraph 3.3.1 (General Environment). However, if any or all of the mathematical models that define the airborne software environment are lengthy, then those models can be documented in appendices, and Paragraph 3.3.1 can merely mention each model and refer to the appropriate appendix for the model description.

4.9 DATA BASE REQUIREMENTS

4.9.1 Purpose of Data Base Requirements

Data base requirements summarize in descriptive and quantitative terms the constants and parameters needed by the airborne software to satisfy its functional and performance requirements. Constants needed by the airborne software may consist of independent constants, aircraft-dependent constants, and mission-dependent constants. An independent constant is a constant that will always have a fixed value such as the earth gravitation constant and the earth rotational rate. These constants are usually considered to be part of the airborne software, and the developing contractor is required to provide them when he delivers the airborne software.

Aircraft-dependent constants are constants that will have different values for each aircraft in which the airborne software is utilized but which will have the same value for every mission. The constants may include, for example, instrument compensation constants and gunnery alignment constants. If these constants are also considered to be part of the airborne software, then the developing contractor is required to provide a set of these constants for each aircraft employing the airborne software.

Mission-dependent constants are constants that will vary from mission-to-mission such as target locations, vehicle mass properties, and vehicle initial position. These are generally not considered to be part of the airborne software, and they may be generated by a separate organization.

All of the known airborne software input constants should be listed and described, and criteria for selecting or methods of generating each constant should be specified. The anticipated range of values for each aircraft-dependent and mission-dependent constant should also be specified in the data base requirements. The values of constants and/or the values of parameters needed to compute the constants may be provided in the airborne software requirements.

Those portions of the data base that are considered to be part of the deliverable airborne software are subject to the same functional and performance requirements as the coding in the airborne software.

Data base requirements may also specify constraints on the data base structure in order to provide for future changes and/or expansion of the data base.

4.9.2 Derivation of Data Base Requirements

Data base requirements are determined by analyzing the interface requirements and the functional requirements. Some of the constants needed by the airborne software will be specified as inputs in the interface requirements. For example, instrument compensation constants may appear in the interface requirements between the airborne software and a calibration tape. Other constants needed by the airborne software may appear as inputs to the functions specified in the functional requirements. The list of constants in the data base requirements may not be the complete list of all the constants that will be needed by the airborne software because the complete list of constants depends on the final design.

The description of each input constant should specify which functions utilize the constant and how the functions utilize the constants.

Criteria for selecting or methods of generating each constant and its anticipated range of values are derived by analyzing how the airborne software functions will utilize the constant and determining the source data on which the constant is based. For example, an instrument compensation constant is based on calibrating the instrument, and its range of values depends on the anticipated values of instrument parameters. A mission-dependent constant is based on mission requirements (target locations, weapons to be employed, etc.), and its range of values is determined by the spectrum of missions that the weapon system is required to perform.

Data base structure requirements are based on anticipated future changes to the data base.

4.9.3 Specification of Data Base Requirements

If the airborne software requirements are documented in MIL-STD 483 format, then the data base requirements should be presented in Paragraph 3.3.2 (System Parameters).

5. DEVELOPMENT STANDARDS AND CONSTRAINTS

The software development standards and constraints specify how the full scale development contractor is to accomplish the following goals during airborne software development: assure compatibility among computer program components; ensure an orderly, systematic development procedure; ensure adequate software documentation; achieve a high quality product; and provide for future modifications and growth to the software. Development standards minimize the life cycle cost of the airborne software by reducing management, checkout and testing costs during software development and by minimizing operation, modification, and maintenance costs after development. These costs savings are attributed to the systematic development procedure and the well documented, high quality software product enforced by the development standards. For large, complex airborne software the development standards may also reduce costs of design and coding because of the orderly development procedure which tends to eliminate costly design and coding errors. The standards also ensure that the airborne software will be compatible with the airborne avionic digital processors and other subsystems.

In some cases, the full scale development contractor is required to prepare development standards as part of the development contract. An advantage to including them in the RFP is that the potential development contractor (bidder) can use the information provided in the development standards to aid him in arriving at an accurate cost estimate for his proposal. The standards in the RFP can also aid the selected development contractor to produce a high quality end product.

Software development standards generally include a software development procedure, a configuration management plan, design standards, programming standards, software testing standards, quality assurance standards, and documentation standards. They should conform with SAMSO Exhibit 73-3 whenever it is applicable.

If the airborne software requirements are documented in MIL-STD 483 format, then most of the development standards and constraints should appear or be referenced in the last subparagraph of paragraph 3.2, called the special requirements section. Development standards are generally included in the RFP by reference to separate software standards documents.

The subsections of this section describe how to derive and specify airborne software development standards and constraints. Table 5-1 summarizes the topics covered in this section and provides a cross reference of applicable government documents.

5.1 SOFTWARE DEVELOPMENT PROCEDURE

The software development procedure should describe each step in the software development process. This portion of the software standards should reflect MIL-STD 1521A as described in the SAE Guidebook for Reviews and Audits.

5.2 CONFIGURATION MANAGEMENT PLAN

The configuration management plan should describe the activities that are to be performed by the airborne software development contractor for configuration management. The plan should reflect MIL-STD 1521A as described in the SAE Guidebook for Reviews and Audits, and it should reflect MIL-STD 480 and 483 as described in the SAE Guidebook for Configuration Management.

5.3 DESIGN STANDARDS

Generally, software design standards require the use of a top-down design procedure; however, in cases where significant portions of the airborne software already exists or when certain lower level algorithms have high technical risk, a combination of top-down and bottom-up design is often recommended. Top down design requires that the design be performed by starting with the top level function and proceeding downward in the design of successively lower level functions. This design approach enhances design traceability, completeness, and comprehensiveness.

Table 5-1. Cross Reference for Section 5

Section Number	Subject	Applicable Government Documents
5.1	Software Development Procedure	MIL-STD 1521A
5.2	Configuration Management Plan	MIL-STD 480 MIL-STD 483 MIL-STD 1521A
5.3	Design Standards	MIL-STD 483 SAMSO Exhibit 73-3
5.4	Programming Standards	SAMSO Exhibit 73-3
5.5	Software Testing Standards	DODD 5000.3
5.6	Quality Assurance Standards	MIL-S 52779 (AD)
5.7	Documentation Standards	MIL-STD 1521A
5.8	Language Requirements	AFR 300-10 DODD 5000.3
5.9	Classified Data Requirements	See Section 2.5
5.10	Testability Requirements	MIL-STD 483
5.11	Expandability Requirements	MIL-STD 483
5.12	Government Furnished Property List	MIL-STD 483
5.13	Media Requirements	MIL-STD 483
5.14	Identification Requirements	AFR 800-14 MIL-STD 483

Top down design is initiated by establishing a functional hierarchy. The top level function in the hierarchy is the overall mission to be performed by the software. Successively lower levels are obtained by breaking down and partitioning the software functions into functions with progressively greater detail. The software functional requirements should be allocated and mapped onto this design hierarchy. The lowest level functions of the design hierarchy should account for all of the software functions, inputs and outputs.

After the functional hierarchy is established, the design of each function can take place, starting with the top level function and proceeding down each successive level of the hierarchy. At each level of design, the higher level function designs should be reviewed and expanded upo Functions, criteria, and design concepts should be evaluated iteratively. Software components, functions performed by each and interactions and interfaces between the software components should be established at each level of the design. Any discrepancies between the software design and the functional requirements should be resolved at each level of design. Criteria, rationale, and tradeoffs used to establish the software design should be recorded.

In addition to requiring top-down design, the software design standards should include design representation standards to promote uniformity, readability, understandability, and maintainability of the airborne software design. The design representation standards should be compatible with the programming language selected for airborne software development, they should be compatible with top-down design, and they should be compatible with the structured programming standards. The SAE Guidebook for Computer Program Documentation Requirements addresses design representation standards and indicates how to amend MIL-STD 483 to allow the use of structured design representations (e.g., HIPO and PDL) as well as flow charts.

5.4 PROGRAMMING STANDARDS

Programming standards promote uniformity, readability, testability, understandability, maintainability, and reliability in the airborne software. When applicable, the programming standards can also contribute to portability of the airborne software between different computers and compatibility with existing and future support software. Standards should be established for each programming language that is to be used in airborne software development. The standards should specify requirements for preface text and inline comments in the coding, and they should specify structured programming rules appropriate to the languages being used in airborne software development. The standards should also address memory utilization, maximum routine size, duty cycle, naming of variables, and modular construction. The primary objectives for programming standards are to minimize life cycle cost and to maximize software reliability (minimizing the probability of coding errors). These objectives are accomplished by ensuring a high quality, understandable (readable), well structured, modular software product, thereby reducing checkout, verification, and validation costs; documentation cost; management costs; and utilization, modification, and maintenance costs. The structured programming rules must be compatible with the flow chart standards as well as the programming languages so that the task of producing code corresponding to the flow charts becomes a straightforward, systematic procedure.

The SAE Guidebook for Computer Program Documentation Requirements and the SAE Guidebook for Quality Assurance address programming standards.

5.5 SOFTWARE TESTING STANDARDS

Software testing standards describe the testing required to demonstrate that the airborne software is a reliable, high quality product that satisfies all the requirements and constraints in the airborne software requirements with high confidence over the spectrum of missions and system parameter variations specified in the requirements. The

standards should also require testing of the airborne software relative to the development and documentation standards and constraints. The testing standards should establish software testing methods and criteria for software certification. The software testing standards should reflect Section 60.4.4 of MIL-STD 483. Software testing is described in the SAE Guidebook for Verification, Validation and Certification and also in the SAE Guidebook for Software Quality Assurance.

If the airborne software requirements are documented in MIL-STD 483 format, then the software testing requirements should appear in Section 4 called Quality Assurance.

5.6 QUALITY ASSURANCE STANDARDS

Quality assurance standards establish quality assurance objectives and describe the activities to be performed to meet those objectives. Software testing discussed in Section 5.5 above is one aspect of quality assurance. The quality assurance standards should also require quality assurance activities during the early phases of software development to ensure that high quality airborne software is developed to minimize the life cycle cost of the weapon system. The SAE Guidebook for Software Quality Assurance describes the activities required for a good quality assurance effort as specified in MIL-S 52779 (AD).

5.7 DOCUMENTATION STANDARDS

The airborne software documentation standards should describe the documents that the development contractor is required to produce. The following airborne software documentation may be required:

- Program Plans
- CPCI Development Specifications
- CPCI Interface Specifications
- CPCI Product Specification (Build-to)
- CPCI Product Specification (As-built)
- CPCI Development Standards
- Software Acceptance Plans
- Test Plans and Procedures
- Test Results
- User's Manuals

The purpose and contents of each of these documents and any other documents that are required should be specified in the documentation standards. The SAE Guidebook for Computer Program Documentation Requirements describes each of the above documents.

5.8 LANGUAGE REQUIREMENTS

Language requirements specify what programming languages (e.g., JOVIAL) and dialects (e.g., J73) are permissible for use during airborne software development. They may also specify under what conditions various languages can be used in different portions of the airborne software. Language requirements may be different for software that executes on different computers in the airborne digital processing subsystem. Language requirements should be based on minimizing the life cycle cost of the weapon system. The programming language may affect the life cycle costs of the airborne software (design, development, verification, validation, and maintenance), the support software (development and maintenance of compilers, assemblers, operating systems, etc.), digital processors (design, manufacture, and maintenance) and other interfacing subsystems.

As mentioned above, selection of programming languages and dialects is based on minimizing the life cycle cost of the weapon system. If a compiler exists for a computer in the avionic digital processing subsystem, requirements to utilize the high level language/dialect that is compatible with that compiler may minimize life cycle cost. However, if that language/dialect is inefficient (in terms of memory utilization), difficult to program, or costly to maintain, it may be less costly overall to modify that compiler or to develop a new compiler. Similarly, if no compiler exists for a computer in the airborne digital processing subsystem, then the cost of developing a compiler for that computer and using its high level language for airborne software development may result in lower life cycle cost than using assembly language for airborne software development. Using a high level language generally reduces software development and maintenance costs (relative to using assembly

language); however, using a high level language also tends to require more computer memory for program storage (thereby increasing computer costs), and it sometimes imposes the added cost of compiler development and/or maintenance. AFR 300-10 specifies computer programming languages that are acceptable for use in software development for the Air Force; however, a waiver of this regulation can sometimes be obtained, for example, on the basis of minimizing life cycle cost.[†] Part of software requirements analysis is to perform comparisons of weapon system life cycle costs corresponding to the various language alternatives so that the most cost effective language can be selected.

If the airborne software requirements are documented in MIL-STD 483 format, then the programming language requirements should appear in the subparagraphs of paragraph 3.1.1.2 as interface requirements between airborne software and the compilers and/or assemblers. If one of the compilers or assemblers to be utilized already exists, it should be identified along with reference to its language description documentation. If a compiler or assembler does not yet exist, the language characteristics should be described, either directly or by reference to appropriate documentation.

5.9 CLASSIFIED DATA REQUIREMENTS

Airborne software may include classified data, classified codes, and/or classified algorithms. Consequently, special handling and storage of portions of the airborne software and its documentation may be required. Classified data requirements specify software design constraints that protect the classified data, i.e., to prevent unauthorized access to the classified information and to prevent alteration (sabotage) of the classified material.

Classified data requirements generally depend on the nature of the classified material, its level of classification, and how it is used and communicated. The requirements are based on security requirements in appropriate security manuals such as those listed in Section 2.5, above. There are often special security manuals for specific weapon systems.

[†] AFR 300-10 specifies the waiver procedure.

5.10 TESTABILITY REQUIREMENTS

Testability requirements specify features in the airborne software design to facilitate efficient testing of the airborne software against its requirements. For example, there may be requirements for output of special data such as intermediate calculation results, and there may be requirements for special interfaces with other subsystem and peripheral equipment so that airborne software input data and output data can be monitored and recorded.

Testability requirements may also be imposed on the airborne software to reduce maintenance (modification, update, and trouble shooting) costs or for quality assurance purposes.

Testability requirements are based on the testing that must be performed to verify that other airborne software requirements are being satisfied. Testing requirements may simply specify that the developing contractor provide whatever airborne software features that are needed to facilitate economical testing against the requirements and to provide for certain maintenance operations. Alternatively, the testability requirements can be more specific by specifying what types of testing must be supported by the testability features or by specifying precisely what data must be output to monitoring equipment such as output tape units, cathode ray tubes, printers, or telemetry equipment. These testability features required of the airborne software can be derived by analyzing the proof of testability for each requirement. The proof of testability consists of defining an economical method of testing the airborne software against each requirement. Airborne software features that are required to conduct these tests should be identified for each method of testing. Hence, the required features needed to test the airborne software against each requirement are identified.

Additional airborne software capabilities needed for maintenance and quality assurance are based on analyzing the maintenance and quality assurance activities that are anticipated after initial operational capability.

Testability requirements may also be based on analyzing the testing methods recommended in the SAE Guidebook for Verification, Validation and Certification. Testability requirements are covered in the SAE Guidebook for Computer Program Maintenance and in the SAE Guidebook for Software Quality Assurance.

Testability requirements to provide data to monitoring and recording devices should be included in the interface requirements section (paragraph 3.1.1), a separate paragraph specifying the interface requirements between the airborne software and each monitoring and recording device. Testability requirements that affect only one airborne software function can be included in the functional requirements subparagraph for that function. All remaining testability requirements should be inserted into the last functional requirements paragraph (Special Requirements).

5.11 EXPANDABILITY REQUIREMENTS

Expandability requirements constrain the airborne software to utilize less than the full capability of the airborne digital processors. They may limit the computer memory utilized by the airborne software, the processor duty cycle (throughput), the input/output interface devices utilized, data bus channels utilized, and the level of usage of other subsystems that interface with the airborne software. The expandability requirements may also specify constraints on the software structure, e.g., modularity, in order to provide for future growth of the airborne software at a minimum cost. Modularity enables one portion (module) of the software to be modified and/or expanded without significantly affecting the rest of the software. The purpose of these requirements is to minimize the cost of future updates and modifications to the weapon system; i.e., the requirements provide for expansion of the airborne software without requiring changes to the airborne digital processor hardware and the interfacing subsystems.

Expandability requirements can be based on providing for identified anticipated improvements to the weapon system, or they can be based on providing for unidentified future improvements (or a combination of both). If identified improvements are anticipated, then the expandability requirements are derived by estimating the additional system software capacity required to provide for the anticipated improvements. This is the amount by which the current budget for the airborne software must differ from the total software capacity of the airborne digital processors and the interfacing subsystems. The portions of the airborne software that will be affected by anticipated improvements should also be determined.

Expandability requirements for unidentified (unknown) future changes to the airborne software are based on past history of the evolutions of similar weapon systems and on minimizing the expected value of life cycle cost. For example, there is a tradeoff between investing dollars into the current weapon system to provide for expandability that may never be needed in this weapon system as opposed to spending dollars later to provide that additional hardware needed for airborne software expansion. Tradeoffs such as this are inexact at best, but they should still be attempted.

It should be noted that spare processing capacity (throughput[†] and storage) should be provided beyond that needed for expandability. Experience shows that software development cost, both initial development and subsequent updates, increases significantly as spare processing capacity goes down. Figure 5-1 depicts how the cost of software development increases as the spare capacity goes down. Limited processor capacity may also necessitate compromises in the airborne software design which produce degraded performance. The amount of spare processing capacity that should be provided beyond that needed for future expansion should be based on minimizing the life cycle cost of the total weapon system, i.e., a compromise between software costs and processing hardware costs.

[†]Throughput is the number of machine level instructions that are executed per second.

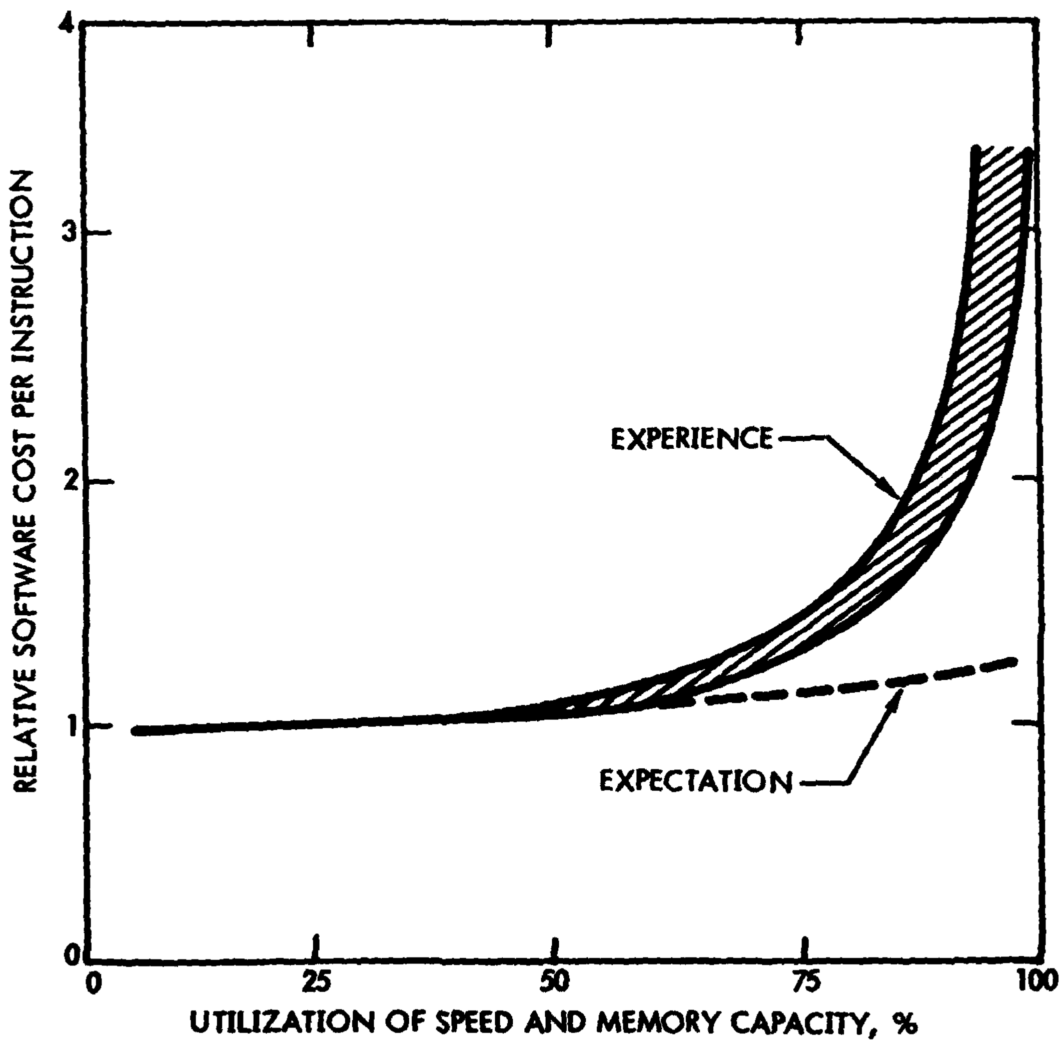


Figure 5-1. Effect of Hardware Constraints On Software Cost

Expandability requirements must be coordinated between the airborne software requirements analysis and the requirements analysis for other subsystems in order to ensure the airborne software can be built within its software budget (expandability constraints) and to ensure that the airborne software together with the other subsystems will provide the required spare capacity and flexibility. The SAE Guidebook for Software Quality Assurance addresses software maintainability.

Expandability requirements should be included as constraints in the interface requirements paragraphs of the airborne software requirements. Each expandability requirement limits the usage of an equipment that interfaces with the airborne software, e.g., an airborne digital computer. The expandability requirement should be included in the interface requirements paragraph for that piece of equipment.

5.12 GOVERNMENT-FURNISHED PROPERTY LIST

The government-furnished property list specifies what facilities and equipment are to be provided to the developing contractor by the government for use in developing and testing the airborne software. The list includes any government furnished software which the airborne software must be designed to incorporate, such as existing CPCI's that are to become part of the new airborne software. The list also includes software, computer hardware, peripheral processing equipment, and other equipment that is to be provided by the government for use in developing and testing the airborne software.

The government-furnished property list should include any available government owned software and equipment whose use may reduce the life cycle costs. The identification of equipment and software that may be useful to the developing contractor is based on the equipment and software used previously in developing and testing software similar to the airborne software, on the analysis of testability of the airborne software and on testability requirements. The proof of testability of each airborne software requirement generally consists of defining an economically feasible method of testing the requirement. Any equipment and software that are

required to perform that testing may be useful to the developing contractor. Of the equipment that may be useful to the developing contractor, any equipment that is government owned and that will be available for use during the airborne software development and testing period should be included on the list of government-furnished equipment. In addition, processing equipment, systems software, compilers, assemblers, and simulation software that could be useful during airborne software development and that are government owned should be candidates for inclusion in the list.

Any existing CPCI that could be usable in developing the airborne software should also be included on the list of government-furnished equipment.

If the requirements are documented in MIL-STD 483 format, then the government-furnished property list should be a subparagraph of the last functional requirements paragraph (Special Requirements). The list should specify the software and equipment by nomenclature, specification number, model number, and associated documentation. The list should be divided into government-provided property and government-loaned property.

5.13 MEDIA REQUIREMENTS

Media requirements specify the physical media through which the airborne software is to be delivered to the Air Force procuring agency. For example, the developing contractor may be required to deliver the airborne software in the form of a magnetic tape, a card deck, a disk, a punched mylar tape, or a combination of the above. The format in which the airborne software is stored in the required media should also be specified. For example, the card punch character set and the tape format (density, parity, and encoding format) should be specified. Source code, as well as object code, may be required and should be specified.

Media requirements are based on the peripheral processing equipment (tape decks, card readers, disk readers, etc.) available for loading the airborne software into the airborne digital processors and on equipment available to other users. The _SAE Guidebook for Configuration Management_ discusses software media.

If the airborne software requirements are documented in MIL-STD 483 format, then the media requirements may appear in Section 5 (Preparation for Delivery).

5.14 IDENTIFICATION REQUIREMENTS

Identification requirements specify what markings will be used on the deliverable items, i.e., computer program media and documentation. The requirements may specify that the identification markings be visually and machine readable and that the markings provide direct correlation between the computer program media (e.g., disks, magnetic tapes, card decks, mylar tapes) and the computer program documentation. The requirements should also specify that a Computer Program Identification Number (CPIN) will be assigned by AFLC to each CPCI in the airborne software before the Critical Design Review (CDR) as specified in AFR 800-14.

The identification requirements may specify criteria for selecting markings, e.g., visually and machine readable, correlation between computer program media and its documentation, unique to each computer program, clear and unambiguous. Alternatively, the identification requirements can specify precisely the markings that must be placed on each deliverable CPCI, e.g., the CPIN. The requirements in MIL-E 5400 may be helpful in deriving identification requirements. The SAE Guidebook for Configuration Management also addresses software and documentation identification.

If the requirements are documented in MIL-STD 483 format, then the identification requirements should appear in Section 5 (Preparation for Delivery).

*U.S.Government Printing Office: 1980 — 657-084/28